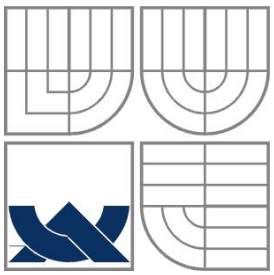
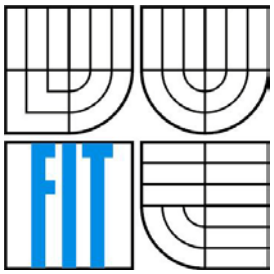


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ROZPOZNÁVÁNÍ VÝRAZU TVÁŘE

FACIAL EXPRESSION RECOGNITION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

BC. JIŘÍ KRÁL

VEDOUCÍ PRÁCE
SUPERVISOR

ING. MICHAL ŠPANĚL

BRNO 2010

Abstrakt

Tato práce představuje jeden z mnoha pohledů na rozpoznávání výrazu lidské tváře. Zaměřuje se na metodu reprezentace obličeje modelem. Metodu AAM vytvářející model vzhledu z modelu tvaru a z modelu textury na základě statistické analýzy. Ukazuje přednosti této reprezentace ve statickém snímku, především pak komplexnost informace, kterou o hledaném obličeji obsahuje. Popisuje princip na AAM založeném rozpoznávání výrazu tváře jako je výběr a kombinace vhodných příznaků z modelu pro následnou klasifikaci výrazu tváře. Porovnává dva přístupy klasifikace výrazu tváře, klasifikaci založenou na LDA a klasifikaci založenou na SVM. Zmíněné metody společně s nezbytnou lokalizací obličeje pomocí metody AdaBoost představují postup automatizovaného rozpoznávače výrazu lidské tváře.

Abstract

Many views to facial expression recognition exist. This work presents one of approaches. Existing methods of human face representation by model are discussed. The AAM method, where final appearance model is created from model of shape and model of texture is proposed. Model of shape and model of texture is created by statistic analysis. Using this representation, an effective method is achieved that is complexity of information for searched face in static image. Choice and combination of suitable features for classification of facial expression is principle for facial expression recognition based on AAM. Two approaches of facial expression classification are compared. Classification based on LDA and classification based on SVM. These methods with necessary face localization using AdaBoost form an automated face recognizer in image.

Klíčová slova

Obličejové rysy, lokalizace obličeje, Active Appearance Models, klasifikace výrazů.

Keywords

Facial features, face localization, Active Appearance Models, expression classification.

Citace

Král Jiří: Detekce obličejových rysů v obraze, diplomová práce, Brno, FIT VUT v Brně, 2010.

Rozpoznávání výrazu tváře

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Michala Španěla. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Jméno Příjmení
Datum

Poděkování

Poděkování patří vedoucímu mé práce Ing. Michalu Španělovi za obětavost a metodické rady při zpracování diplomové práce.

Obsah

Obsah 1
1 Úvod 3
2 Detekce objektů 4
2.1 Příznaky 4
2.2 Integrální obraz 5
2.3 Učení klasifikačních funkcí 5
2.4 Kaskádová struktura 7
3 Reprezentace obličeje pomocí AAM 8
3.1 Model tvaru 8
3.2 Model textury 13
3.3 Kombinovaný model 16
3.4 Active Appearance Models 18
4 Statistické metody a klasifikace 21
4.1 Analýza hlavních komponent 21
4.2 Lineární diskriminační analýza 21
4.3 SVM 22
5 Přehled existujících metod 23
5.1 Lokalizace obličeje 23
5.2 Reprezentace obličeje 24
5.3 Klasifikace výrazů 25
6 Návrh rozpoznávače výrazu tváře 26
6.1 Lokalizace obličeje 26
6.2 Nalezení AAM modelu 27
6.3 Extrakce příznaků 29
6.4 Klasifikace výrazů tváře 32
7 Implementace rozpoznávače výrazu 34
7.1 Použité nástroje 34
7.2 Struktura programu 35
8 Experimentální výsledky 36
8.1 Trénování modelu 36
8.2 Optimalizace parametrů 36
8.3 Klasifikace výrazů tváře 37
9 Závěr 44
Literatura 46

A Příloha 47
A.1 Popis programu Build 47
A.2 Popis programu Lda 47
A.3 Popis programu Fit 48

1 Úvod

Výrazy lidské tváře slouží k vyjádření emocí člověka. Ještě před vznikem řeči se výrazové prostředky společně s různými gesty používali ke komunikaci mezi lidmi. Výraz tváře, který vyjadřuje určitý pocit člověka, může být obtížné popsat i za pomocí mnoha slov. Díky velkým nárůstům výpočetní výkonnosti se v poslední době začínají obličejové rysy prosazovat i v počítačovém zpracování. Jednou z hojně rozšířených oblastí, kde se využívá obličejových rysů, jsou snímací zařízení. Příkladem mohou být fotoaparáty, které pořídí snímek na základě detekce úsměvu.

Další oblastí, kde se mohou začít výrazy obličeje prosazovat je komunikace člověka s počítačem. Časem by tento přirozený druh komunikace společně s hlasem mohl nahradit standardní periférie jako je klávesnice a myš. Zatímco hlasové ovládání se začíná pomalu integrovat do různých počítačových aplikací, výraz obličeje zůstává zatím v ústraní.

Rozpoznávání výrazu lidské tváře je komplexní a náročná úloha především kvůli velké variabilitě a individuálnímu způsobu vyjádření emocí jednotlivých osob. Při pořizování snímku obličeje je nutné zohlednit vlivy jako je osvětlení obličeje, úhel pořízení obličeje, natočení hlavy, ale i různé doplňky, jako jsou brýle a vousy.

V rámci této práce se pokusím ukázat možnosti analýzy a zpracování obrazu obsahující lidský obličej včetně jeho výrazových charakteristik. Především se zaměřím na reprezentaci obličeje s vhodným výběrem takových obličejových charakteristik, které lze využít ke klasifikaci výrazu tváře. Samozřejmě budou zmíněny i metody lokalizace obličeje, které jsou pro další postup nezbytné.

Úvodní druhá kapitola se zabývá právě lokalizací obličeje, konkrétně rychlým a robustním přístupem založeným na strojovém učení AdaBoost. Třetí, poměrně rozsáhlá kapitola se již věnuje reprezentaci obličeje. Metoda AAM, která vytváří pro hledaný obličej odpovídající reprezentaci modelem. Čtvrtá kapitola shrnuje metody pro možnou klasifikaci výrazů. Pátá kapitola je jakýmsi průřezem v současnosti používaných systémů pro lokalizaci, reprezentaci a klasifikaci. Šestá kapitola se již věnuje samotnému návrhu rozpoznávače výrazu obličeje založeného na AAM. Budou v ní zmíněny jednotlivé možnosti klasifikace, úpravy a vylepšení stávajících metod. Sedmá kapitola shrnuje a analyzuje dosažené výsledky implementovaného systému.

Nyní nezbývá nic jiného, než přistoupit k samotné diplomové práci. Prací, která nepřímo navazuje na předchozí semestrální projekt, jenž mi dal dostatečné znalosti o problematice detekce obličejových rysů a jednotlivých metodách, které mohu dále rozvíjet v této práci.

2 Detekce objektů

Tato kapitola popisuje metodu detekce objektů vycházející z práce P. Violy a M. Jonese [1]. Jimi navržené algoritmy poskytují robustní a extrémně rychlé detekce objektů. I když tato metoda lze prakticky použít pro detekci libovolných objektů, je konstruována pro detekci obličejů z předního pohledu.

Je použita nová reprezentace obrazu nazývána integrální obraz. Detekční systém nepracuje přímo s intenzitami pixelů, ale s příznaky nazývané Haarovy příznaky. Díky integrálnímu obrazu lze tyto příznaky velmi rychle vyhodnocovat v různých velikostech obrazu. Výpočet integrálního obrazu nezabere více, jak pár operací na pixel. Po výpočtu integrálního obrazu lze jakýkoliv příznak o libovolné velikosti a na libovolné pozici vypočítat v konstantním čase.

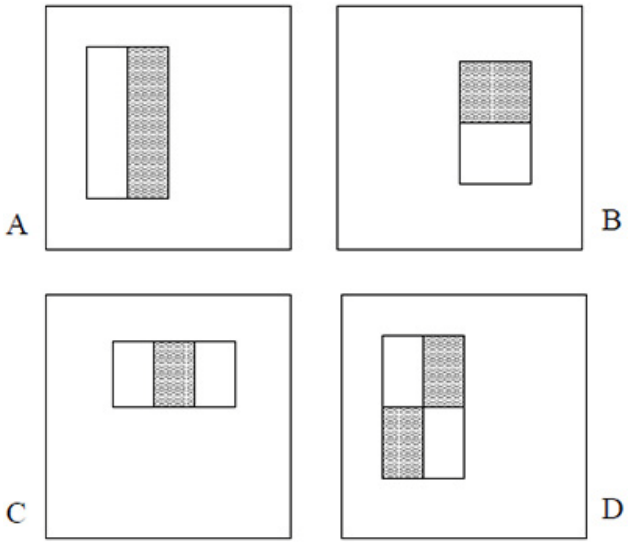
Konstrukce klasifikátoru spočívá ve výběru malého množství významných příznaků. Příznaků bývá podstatně více, než pixelů v obraze. Pro zajištění rychlé klasifikace musí proces učení vyloučit většinu příznaků a ponechat pouze malou množinu významných příznaků.

Následně se kombinují postupně více komplexní klasifikátory do kaskády, která významně zvyšuje rychlost detektoru zaměřením se na slibné oblasti obrazu. Komplexnější zpracování je pak rezervováno pouze pro tyto slibné oblasti. Musí být ale splněno, že všechny nebo téměř všechny instance daného objektu budou vybrány.

2.1 Příznaky

Detekční procedura je založena na hodnotách jednoduchých příznaků. Hlavním důvodem, proč jsou použity příznaky místo pixelů je, že příznaky umí kódovat informace, které je obtížné se naučit z konečné množiny trénovacích dat. Dalším důvodem je, že systémy založené na příznacích pracují výrazně rychleji, než běžné systémy založené na pixelech.

V detektoru jsou použity tři druhy příznaků (obr. 2.1). Hodnota příznaku je dána rozdílem sum jednotlivých obdélníkových oblastí. Základní rozlišení detekčního okna je 24×24 pixelů, úplná množina má 180 000 obdélníkových příznaků.



Obr. 2.1: Příklad obdélníkových příznaků v detekčním okně. Sumy pixelů ležících v bílých obdélnících jsou odečteny ze sumy pixelů ležících v šedých obdélnících. Dvou-obdélníkový příznak je zobrazen v (A) a v (B). (C) ukazuje tří-obdélníkový příznak a (D) ukazuje čtyř-obdélníkový příznak (P. Viola a M. Jones, 2001).

2.2 Integrální obraz

Obdélníkové příznaky mohou být velmi rychle vypočítány za použití pomocné reprezentace obrazu, která se nazývá integrální obraz. Integrální obraz na pozici x, y obsahuje sumu pixelů nad a nalevo od pozice x, y:

(2.1)

$$ii(x,y) = \sum_{x' \leq x, y' \leq y} i(x', y')$$

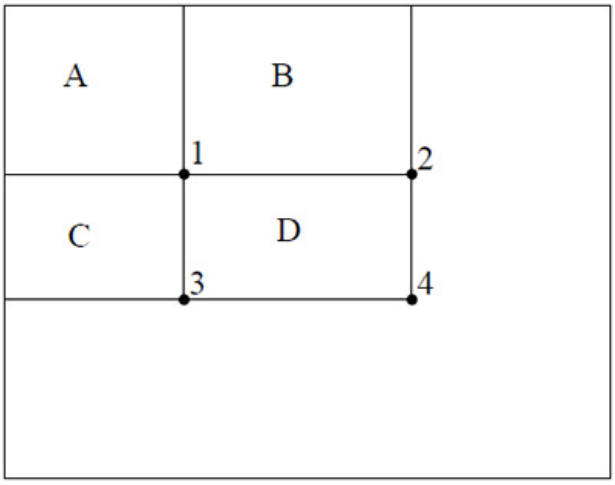
kde $ii(x,y)$ je integrální obraz a $i(x,y)$ je originální obraz. Použije se následující rekurzivní zápis:

(2.2)

$$s(x,y) = s(x,y-1) + i(x,y)$$
$$ii(x,y) = ii(x-1,y) + s(x,y)$$

kde $s(x,y)$ je kumulativní suma, $s(x,-1) = 0$ a $ii(-1,y) = 0$. Integrální obraz může být spočítán jedním průchodem originálního obrazu.

Za použití integrálního obrazu lze vypočítat obdélníkovou sumu ze čtyř polí (obr. 2.2). Z toho je zřejmé, že rozdíl mezi dvěma obdélníkovými sumami lze spočítat z osmi polí. Ale v případě výše zmíněného dvou-obdélníkového příznaku pro výpočet stačí pouze šest polí, protože příznak zahrnuje sousední obdélníkové sumy. V případě tří-obdélníkového příznaku pak pro výpočet stačí osm polí a devět polí v případě čtyř-obdélníkového příznaku.



Obr. 2.2: Suma pixelů uvnitř obdélníku D může být vypočítána ze čtyř polí. Hodnota integrálního obrazu na pozici 1 je suma pixelů v obdélníku A. Hodnota na pozici 2 je $A + B$, na pozici 3 je $A + C$ a na pozici 4 je $A + B + C + D$. Suma uvnitř D může být spočítána jako $4 + 1 - (2 + 3)$ (P. Viola a M. Jones, 2001).

2.3 Učení klasifikačních funkcí

Je-li dána množina příznaků a trénovací množina pozitivních a negativních obrazů, pak lze použít některý z přístupů strojového učení pro určení klasifikační funkce. V tomto systému je použita metoda AdaBoost.

Pro připomenutí je 180 000 obdélníkových příznaků souvisejících s každým oknem obrazu a je jich mnohem více než pixelů v obraze. I kdyby byl každý příznak vypočítán velmi efektivně, tak

přesto výpočet kompletní množiny by byl neúměrně nákladný. V této hypotéze je předpokládáno, že velmi malý počet příznaků může být zkombinován do efektivního klasifikátoru. Hlavním úkolem je tedy nalézt tyto příznaky.

Pro dosažení tohoto cíle je navržen tzv. slabý učící algoritmus pro výběr jednoho obdélníkového příznaku, který nejlépe oddělí pozitivní a negativní příklady. Pro každý příznak slabý učící algoritmus určí optimální práh klasifikační funkce tak, aby minimum příkladů bylo chybně klasifikováno. Slabý klasifikátor $h_j(x)$ tak obsahuje příznak f_j a práh θ_j a paritu p_j:

$$h_j(x) = \begin{cases} 1 & \text{if } p_j f_j(x) < p_j \theta_j \\ 0 & \text{jinak} \end{cases} \quad (2.3)$$

kde x je okno obrazu o velikosti 24×24 pixelů. V tab. 2.1 je shrnut AdaBoost proces.

- Je dán příklad obrazu $(x_1, y_1), \ldots, (x_n, y_n)$, kde $y_i = 0{,}1$ pro negativní respektive pozitivní příklady.
- Inicializace vah $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ pro $y_i = 0{,}1$, kde m a l je počet negativních respektive pozitivních příkladů.
- Pro $t = 1, \ldots, T$:
    1. Normalizace vah,

       $w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$

       tak, že w_t je rozložení pravděpodobnosti.
    2. Pro každý příznak j je trénován klasifikátor h_j, který je vyhrazen pro použití s jedním příznakem. Chyba je vypočítána s ohledem na w_t, $\epsilon_j = \sum_i w_i \left| h_j(x_i) - y_i \right|$.
    3. Zvolí se klasifikátor h_t s nejmenší chybou ϵ_t.
    4. Aktualizace vah:

       $w_{t+1,i} = w_{t,i} \beta_t^{1-\epsilon_i}$

       kde $\epsilon_i = 0$, jestliže příklad x_i je klasifikován korektně, jinak $\epsilon_i = 1$ a $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.
- Konečný silný klasifikátor je:

  $$h_j(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{jinak} \end{cases}$$

  kde $\alpha_t = \log \frac{1}{\beta_t}$.

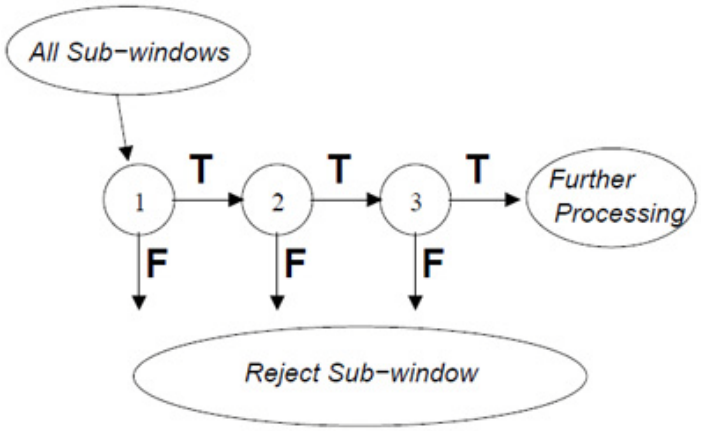
Tab. 2.1: AdaBoost algoritmus pro učení klasifikátoru. V každém kole boostingu je vybrán jeden příznak ze 180 000 potenciálních příznaků.

2.4 Kaskádová struktura

Zde je popsán algoritmus pro sestavení kaskády, která zvyšuje detekční schopnost a radikálně snižuje výpočetní čas. Jednodušší klasifikátory jsou použity pro odmítnutí většiny oken před použitím komplexnějších klasifikátorů, které zase slouží ke snížení falešně pozitivního poměru. Celkový tvar detekčního procesu je degenerativní rozhodovací strom, který se nazývá kaskáda (obr. 2.3). Pozitivní výsledek z prvního klasifikátoru spustí výpočet na druhém klasifikátoru, který má být upravený pro dosažení velmi vysokého detekčního poměru. Pozitivní výsledek z druhého klasifikátoru spustí třetí klasifikátor atd.. Negativní výsledek na jakémkoliv klasifikátoru vede k okamžitému odmítnutí okna.

Struktura kaskády reflektuje ten fakt, že uvnitř jednoho obrazu je převážná většina oken negativních. Kaskáda se pokouší v počátečních stavech odmítnout tolik negativních oken kolik je jen možné. Zatímco pozitivní okna budou spouštět výpočet každého dalšího klasifikátoru v kaskádě.

Podobně jako u rozhodovacích stromů následující klasifikátory budou trénovány na příkladech, které prošly přes všechny předchozí stupně kaskády. Následkem toho druhý klasifikátor obličeje je náročnější úlohou než první klasifikátor. Příklady, které prošly přes první stav jsou náročnější než typické příklady.



Obr. 2.3: Schéma detekční kaskády. Řada klasifikátorů aplikovaných na každé okno. Inicializační klasifikátor eliminuje velké množství negativních příkladů s velmi rychlým zpracováním. Následující vrstvy eliminují další negativní okna, ale vyžadují doplňkové výpočty. Po několika stavech zpracování může být počet oken radikálně zredukován. Další zpracování může být pojato jako přídavné stavy kaskády (jako v popisovaném detekční systému) nebo jako alternativní detekční systém (P. Viola a M. Jones, 2001).

3 Reprezentace obličeje pomocí AAM

Tato kapitola se bude zabývat již samotnou reprezentací obličeje v obraze. K tomu je použita metoda pro reprezentaci modelem AAM (Active Appearance Models), budou podrobně popsány hlavní části metody.

3.1 Model tvaru

Tato kapitolka popisuje techniky, publikované v prácí M. Stegmanna [3], pro pochopení statistického modelu tvaru používaného v AAM. Základem je práce s tvarem jako množinou význačných bodů (landmarks). V následujících kapitolách je demonstrováno, jak mohou být různé tvarové variace efektivně modelovány za použití PCA.

Tvar a význačné body

V prvé řadě by mělo být vysvětleno, co chápat pod pojmem *tvar*. To výstižně popisují následující definice:

„Soubor odpovídajících hraničních bodů."

„Charakteristické uspořádání povrchu objektu daného konturou."

„Nějaké odlišení objektu z okolí pomocí jeho obrysu."

Pro dále zmíněné postupy je především podstatná následující definice:

„Tvar je všechno, co zůstane, když umístění, měřítko a rotace budou z objektu odfiltrovány."

To znamená, že tvar je invariantní vůči euklidovským transformacím.

Jednou možností popisu tvaru je umístěním konečného počtu bodů na jeho obrysu. Těmto bodům se říká význačné body:

„Význačný bod na každém objektu je bod, pomocí něhož se určují shody a rozdíly mezi objekty."

Matematická reprezentace n-bodového tvaru v k dimenzích by mohla spojit všechny dimenze do kn-vektoru. Bude se pracovat pouze s tvary ve 2D, proto $k = 2$. Vektorová reprezentace tvaru v rovině bude vypadat následovně:

$$\mathbf{x} = [x_1, x_2, \dots, x_n, y_1, y_2, \dots, y_n]^{\mathrm{T}} \tag{3.1}$$

Zarovnání tvaru

Aby se dosáhlo skutečné tvarové reprezentace, musí se podle výše zmíněné definice z objektu odfiltrovat efekty umístění, měřítka a rotace. To se provede ustanovením referenčních souřadnic – umístění, měřítko a rotace, do kterých budou všechny tvary zarovnány. Procedura zarovnání do referenčních souřadnic se nazývá Prokrustova analýza. Pro pochopení a ovládání množiny tvarů z nějaké třídy objektů je uveden termín *tvarový prostor*:

„Tvarový prostor je množina všech možných tvarů objektu.“

Nechť k označuje Euklidovskou dimenzi a n označuje počet význačných bodů, dimenze tvarového prostoru je pak dána:

(3.2)

$$M = kn - k - 1 - \frac{k(k-1)}{2}$$

Původně bylo kn dimenzí, ale odfiltrováním translace se odstraní k dimenzí, odfiltrováním měřítka 1 dimenze a odfiltrováním rotace $1/2k(k-1)$ dimenzí.

Prokrustovo měření vzdáleností

Prokrustova vzdálenost je založena na metodě nejmenších čtverců, tudíž vyžaduje, aby počet význačných bodů tvarů byl $1-1$. Určení Prokrustovy vzdálenosti mezi dvěma tvary vyžaduje čtyři kroky:

1. Vypočítat těžiště všech tvarů.
2. Změnit měřítko všech tvarů pro dosažení shodné velikosti.
3. Zarovnání těžišť dvou tvarů do referenční pozice.
4. Zarovnání tvarů podle rotace.

Matematicky čtvercová Prokrustova vzdálenost mezi tvary $\mathbf{x}_1$ a $\mathbf{x}_2$ je suma čtverců nad vzdáleností dvou bodů:

(3.3)

$$P_d^2 = \sum_{j=1}^{n} \left[\left(x_{j_1} - x_{j_2}\right)^2 + \left(y_{j_1} - y_{j_2}\right)^2 \right]$$

Těžiště tvaru může být dáno jako střed ze všech význačných bodů:

(3.4)

$$(\bar{x}, \bar{y}) = \left(\frac{1}{n}\sum_{j=1}^{n} x_j \, , \frac{1}{n}\sum_{j=1}^{n} y_j \right)$$

Pro vykonání kroku 2 je třeba ustanovit metriku pro velikost tvaru:

Metrika velikosti tvaru je kladná, reálná funkce z vektoru tvaru, která splňuje následující vlastnost:

(3.5)

$$\mathrm{S}(a\mathbf{x}) = \mathrm{S}a(\mathbf{x})$$

Následně je jako metrika velikosti použito Frobeniusovo měřítko:

(3.6)

$$S(\mathrm{x}) = \sqrt{\sum_{j=1}^{n} \left[\left(x_j - \bar{x}\right)^2 + (y_j - \bar{y})^2 \right]}$$

Pro odfiltrování rotačního efektu je použita technika SVD (Singular Value Decomposition):

1. Uspořádat velikostně a pozičně zarovnané tvary $\mathbf{x}_1$a $\mathbf{x}_2$ do matice $n \times k$.
2. Vypočítat SVD, $\mathbf{UDV}^{\mathrm{T}}$ z $\mathbf{x}_1^{\mathrm{T}}\mathbf{x}_2$.
3. Pak rotační matice potřebná pro optimální zarovnání $\mathbf{x}_1$na $\mathbf{x}_2$ je $\mathbf{VU}^{\mathrm{T}}$**:**

(3.7)

$$\mathrm{VU}^{\mathrm{T}} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

Alternativou je varianta založená na Prokrustově vzdálenosti, provádějící zarovnání minimalizováním $|\mathrm{T}(\mathbf{x}_1) - \mathbf{x}_2|^2$, kde T v Euklidovském případě je:

(3.8)

$$\mathrm{T}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{bmatrix} a & -b \\ b & a \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

Výraz $|\mathrm{T}(\mathbf{x}_1) - \mathbf{x}_2|^2$ je pak jednoduše rozlišené (a, b, t_x, t_y). Je tak nalezeno řešení pro zarovnání za použití afinních transformací. Nicméně tato transformace mění aktuální tvar.

Zarovnání množiny tvarů

Pro zarovnání množiny tvarů se použije jednoduchý iterativní přístup:

1. Zvolit první tvar jako odhad průměrného tvaru.
2. Ostatní tvary zarovnat na průměrný tvar.
3. Přepočítat odhad průměrného tvaru.
4. Pokud se odhad tvaru změnil, pokračovat krokem 2.

Konvergence je deklarována, když během jedné iterace nedojde k významné změně odhadu průměrného tvaru.

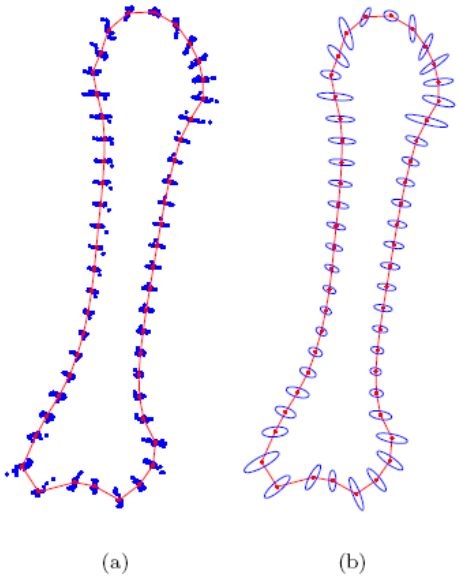
Dále zbývá spočítat první odhad průměrného tvaru. Ve většině případů se používá Prokrustův průměrný tvar:

(3.9)

$$\bar{\mathbf{x}} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_i$$

kde N označuje počet tvarů.

Výsledné tvarové zarovnání může být zobrazeno jako rozptýlené body na obr. 3.1 (a), kde plná čára zobrazuje průměrný tvar. Toto se nazývá bodově distribuovaný model (Point Distribution Model - PDM) všech tvarů. Jak provádět změny modelu v rámci PDM bude popsáno v následující kapitole. Pro jasnější vyjádření bodové variace přes všechny tvary je použita aproximace elipsou pro každý průměrný model daného bodu (viz obr. 3.1 (b)).

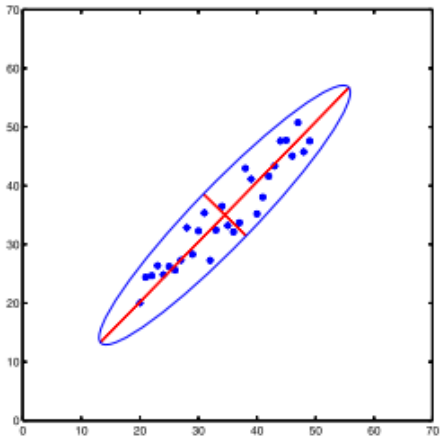


Obr. 3.1: (a) PDM se všemi zarovnanými tvary. (b) Aproximace elipsou pro jednotlivé bodové distribuce z obrázku (a) (Stegmann, 2000).

Modelování tvarových variací

V předchozí kapitole byly zmíněny definice tvaru a způsoby zacházení s tvary. V této kapitole bude demonstrováno, jak lze konzistentně a efektivně popsat tvarové variace.

Základem je klasická statistická metoda, tedy PCA. PCA dává nové osy seřazené podle jejich variability. Na obr. 3.2 jsou zobrazeny dvě hlavní osy dvou-dimenzionální datové množiny, jejichž délky vyjadřují rozsah dat.



Obr. 3.2: Hlavní osy ve 2D (Stegmann, 2000).

V příkladu na obr. 3.2 by se mohlo dosáhnout snížení počtu dimenzí odstraněním druhé hlavní osy a následnou vizualizací bodů ortogonální projekcí do první (delší) osy.

V praxi je PCA vykonávána jako analýza (eigenanalysis) z kovarianční matice ze zarovnaných tvarů.

Je předpokládáno, že množina tvarů představuje nějakou elipsoidní strukturu, z níž může být střed odhadnut jako:

(3.10)

$$\bar{\mathbf{x}} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_i$$

Je evidentní, že takovýto odhad je ekvivalentní odhadu průměrného tvaru.

Maximum likelihood (ML) odhad z kovarianční matice pak může být dán takto:

(3.11)

$$\mathbf{\Sigma}_s = \frac{1}{N}\sum_{i=1}^{N} (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^{\mathrm{T}}$$

Hlavní osy z $2n$-dimenzionálního tvaru elipsoidu jsou dány jako vlastní vektory $\mathbf{\Phi}_s$ z kovarianční matice.

(3.12)

$$\mathbf{\Sigma}_s \mathbf{\Phi}_s = \mathbf{\Phi}_s \mathbf{\Lambda}_s$$

kde $\mathbf{\Lambda}_s$ označuje matici vlastních čísel (eigenvalues):

(3.13)

$$\mathbf{\Lambda}_s = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_{2n} \end{bmatrix}$$

odpovídající vlastním vektorům ve sloupcích matice $\mathbf{\Phi}_s$:

(3.14)

$$\mathbf{\Phi}_s = [\phi_1 \quad \cdots \quad \phi_{2n}]$$

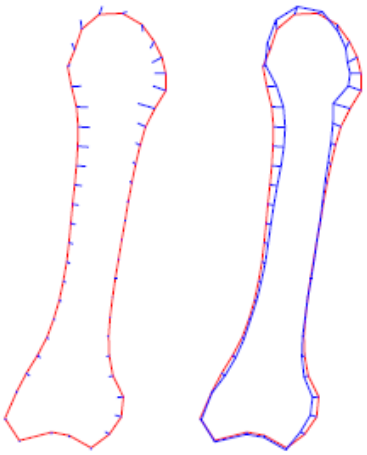
Jednotlivé instance tvaru pak mohou být generovány deformováním průměrného tvaru lineárními kombinacemi s vlastními vektory:

(3.15)

$$\mathbf{x} = \bar{\mathbf{x}} + \mathbf{\Phi}_s \mathbf{b}_s$$

kde $\mathbf{b}_s$ jsou parametry modelu tvaru. V podstatě bodová nebo uzlová reprezentace tvaru byla tímto nahrazena modální reprezentací, kde módy jsou seřazeny podle jejich deformační energie (například procento všech tvarových variací, které lze daným módem pokrýt).

Vlastní vektory jsou množinou vektorů posunutí, kolem kterých je průměrný tvar deformován. Na obr. 3.3 (a) je první vlastní vektor zobrazen na průměrném tvaru. Výsledná deformace z průměrného tvaru je pak na obr. 3.3 (b).



(a) (b)

Obr. 3.3: (a) Průměrný tvar a deformační vektory z prvního vlastního vektoru. (b) Průměrný tvar, deformační vektory z prvního vlastního vektoru a deformovaný tvar (Stegmann, 2000).

Ještě zbývá určit, kolik módů je potřeba udržet. To vede ke kompromisu mezi přesností a kompaktností modelu. Nicméně to zabezpečí vyšší odolnost vůči šumu. Může být ukázáno, že variace kolem osy odpovídají i-tému vlastnímu číslu λ_i. Takto pro udržení p procent variací v trénovací množině je zvoleno t módů:

(3.16)

$$\sum_{i=1}^{t} \lambda_i \geq \frac{p}{100} \sum_{i=1}^{2n} \lambda_i$$

3.2 Model textury

Pro vyjádření kompletního modelu vzhledu (AAM) nestačí uvažovat pouze tvar. Tvar je definovaný odvozením ze znalosti sousedních pixelů. Musí být také zahrnuta informace, představující pixely samotné, tedy jejich hodnoty.

V případě tvaru data byla získána přímo, protože význačné body ve vektoru tvaru představovali data samotná. V případě textury je potřeba nějaká metoda, která seskupí texturní informace mezi význačnými body, například obrazový warping. Existuje několik způsobů, podle [3] bude použit po částech afinní warping založený na Delaunayho triangulaci. Pro získání texturní informace z trénovací množiny, bude každý tvar postupně warpován do referenčního tvaru a deformovaná textura pak bude uložena. Dále je potřeba provést fotometrickou normalizaci získané textury a to odstraněním vlivů globálních lineárních změn v intenzitě pixelů. Analýza textury je podobná analýze tvaru.

Texturový objekt

V počítačové grafice je termín textura přímo závislý na mapování pixelů do virtuálního 2D nebo 3D povrchu. Z toho lze odvodit následující definici:

„Textura je intenzita pixelů na příslušném objektu.“

Matematická reprezentace textury daného objektu je dána jako vektor:

(3.17)

$$\mathbf{g} = [g_1, g_2, \dots, g_m]^{\mathrm{T}}$$

kde m označuje počet pixelů na povrchu objektu.

Obrazový warping

Obrazový warping je jednoduchá metoda transformující jednu prostorovou konfiguraci obrazu do jiné. Proto i jednoduchá translace může být chápána jako obrazový warping. Formálně $\mathbf{I} \in \mathbb{R}^k \rightarrow \mathbf{I}' \in \mathbb{R}^k$, v rovinném případě $k = 2$.

AAM je založena na význačných bodech, proto bude použita taková třída z metod obrazového warpingu, kde dochází k mapování z jedné libovolné bodové množiny $\{\mathbf{x}_1 \dots \mathbf{x}_n\}$ do jiné $\{\mathbf{x}'_1 \dots \mathbf{x}'_n\}$, kde každý bod je reprezentován jako $\mathbf{x} = [x, y]^{\mathrm{T}}$. Formálně zapsáno jako souvislý vektor hodnot mapovaný funkcí takovou, že:

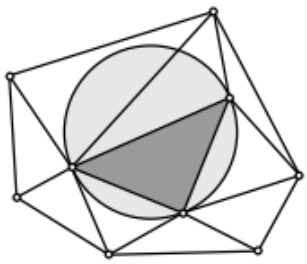
(3.18)

$$\mathbf{f}(\mathbf{x}_i) = \mathbf{x}'_i \quad \forall \quad i = 1 \dots n$$

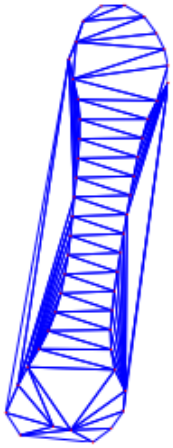
Po částech afinní warping

Nejjednodušší konstrukce na n bodech založeného warpingu je předpoklad, že funkce $\mathbf{f}$ je lokálně lineární. Jeden z přístupů je rozdělit konvexní obálku body za použití vhodné triangulace, jako je například Delauneyho triangulace.

Delauneyho triangulace propojuje nepravidelně rozmístěné body pomocí sítě trojúhelníků, kde každý trojúhelník musí splňovat Delauneyho vlastnosti. To znamená, že pro každý trojúhelník je dána opsaná kružnice a ta neobsahuje žádné další body, než ty, co tvoří vrcholy jejího trojúhelníku (viz obr. 3.4). Delauneyho triangulace průměrného tvaru z předchozí kapitoly je na obr. 3.5. Poznamenejme, že kvůli konkávnímu tvaru jsou trojúhelníky produkovány i vně tvaru.



Obr. 3.4: Opsaná kružnice trojúhelníku splňující Delauneyho vlastnosti (Stegmann, 2000).



Obr. 3.5: Delauneyho triangulace průměrného tvaru (Stegmann, 2000).

Warping je nyní realizován aplikováním triangulační sítě s první bodové množiny **I** do druhé bodové množiny **I´**. Každý bod každého trojúhelníka z první bodové množiny může být jednoznačně namapován do odpovídajícího trojúhelníka druhé bodové množiny a to za pomocí afinních transformací, které v podstatě obsahují změnu velikosti, posunutí a úhlové natočení. Jestliže $\mathbf{x}_1, \mathbf{x}_2$ a $\mathbf{x}_3$ označuje vrcholy trojúhelníka v **I**, nějaký vnitřní bod tohoto trojúhelníka může být zapsán jako superpozice:

(3.19)

$$\begin{aligned}\mathbf{x} &= \mathbf{x}_1 + \beta(\mathbf{x}_2 - \mathbf{x}_1) + \gamma(\mathbf{x}_3 - \mathbf{x}_1) \\ &= \alpha\mathbf{x}_1 + \beta\mathbf{x}_2 + \gamma\mathbf{x}_3\end{aligned}$$

Tak $\alpha = 1 - (\beta + \gamma)$ dává $\alpha + \beta + \gamma = 1$. Pro omezení **x** uvnitř trojúhelníku musí platit $0 \leq \alpha, \beta, \gamma \leq 1$. Warping je nyní dán za použití relativní pozice uvnitř troúhelníku určené pomocí α, β a γ na trojúhelníku v **I´**:

(3.20)

$$\mathbf{x}' = \mathbf{f}(\mathbf{x}) = \alpha\mathbf{x}'_1 + \beta\mathbf{x}'_2 + \gamma\mathbf{x}'_3$$

Z daných tří bodů trojúhelníku je triviální určit α, β a γ a to řešením systému dvou lineárních rovnic daných z rovnice (3.19) a známého bodu, $\mathbf{x} = [x, y]^{\mathrm{T}}$:

(3.21)

$$\begin{aligned}\alpha &= 1 - (\beta + \gamma) \\ \beta &= \frac{yx_3 - x_1y - x_3y_1 - y_3x + x_1y_3 + xy_1}{-x_2y_3 + x_2y_1 + x_1y_3 + x_3y_2 - x_3y_1 - x_1y_2} \\ \gamma &= \frac{xy_2 - xy_1 - x_1y_2 - x_2y + x_2y_1 + x_1y}{-x_2y_3 + x_2y_1 + x_1y_3 + x_3y_2 - x_3y_1 - x_1y_2}\end{aligned}$$

V pseudo-kódu může být po částech afinní warping zapsán takto:

1. Pro každý pixel $\mathbf{x} = [x, y]^{\mathrm{T}}$ uvnitř konvexní obálky z $\{\mathbf{x}'_1 \ldots \mathbf{x}'_n\}$.
    2. Určit trojúhelník t, kterému **x** náleží.
    3. Nalézt relativní pozici **x** uvnitř t za pomocí (3.21).
    4. Použít (3.20) pro získání pozice uvnitř t'.
    5. Nastavit $\mathbf{I}'(\mathbf{x}) = \mathbf{I}(\mathbf{f}(\mathbf{x}))$.

6. Konec.

Naivní řešení kroku 2 je procházet postupně všechny trojúhelníky dokud není splněno $0 \leq \alpha, \beta, \gamma \leq 1$.

Modelování texturních variací

Jedná se o podobný postup jako pro modelování tvarových variací. Pro texturní variace se opět využije PCA přístup. Akorát na rozdíl od modelování tvaru, zde se bude pracovat místo s význačnými body s body textury, kterých bývá podstatně více.

Maximum-likelihood odhad průměrné textury z N normalizovaných textur je dán jako:

(3.22)

$$\bar{\mathbf{g}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{g}_i$$

Maximum-likelihood odhad kovarianční matice pak může být zapsán jako:

(3.23)

$$\Sigma_g = \frac{1}{N}\sum_{\mathrm{i}=1}^{N}(\mathbf{g}_i - \bar{\mathbf{g}})(\mathbf{g}_i - \bar{\mathbf{g}})^{\mathrm{T}}$$

Hlavní osy z m-dimenzionálních bodových shluků z textur jsou nyní dány jako vlastní vektory $\boldsymbol{\Phi}_g$ z kovarianční matice.

(3.24)

$$\Sigma_g \boldsymbol{\Phi}_g = \boldsymbol{\Phi}_g \boldsymbol{\Lambda}_g$$

kde $\boldsymbol{\Lambda}_g$ je diagonální matice vlastních čísel. Instance textury pak mohou být generovány deformováním průměrné textury lineárními kombinacemi s vlastními vektory:

(3.25)

$$\mathbf{g} = \bar{\mathbf{g}} + \boldsymbol{\Phi}_g \mathbf{b}_g$$

kde $\mathbf{b}_g$ jsou parametry modelu textury.

3.3 Kombinovaný model

Náplní této kapitoly je způsob, jakým sjednotit prezentované modely tvaru a textury do jednoho kompletního a kompaktního modelu vzhledu. Je také ukázáno, jak tato reprezentace modelu může být dále sjednocena a zkomprimována snížením počtu parametrů modelu (vlastní módy).

Kombinování modelů tvaru a textury

V předchozích kapitolkách bylo vidět, že instance objektu mohou být konstruovány za pomocí dvou množin parametrů modelu a to parametrů modelu tvaru $\mathbf{b}_s$ a parametrů modelu textury $\mathbf{b}_g$. Pro odstranění korelací mezi parametry modelu tvaru a textury a pro vytvoření kompaktnější reprezentace

modelu je PCA vykonáno na zřetězených parametrech tvaru a textury **b** z trénovací množiny pro získání kombinovaných parametrů modelu **c**:

(3.26)

$$\mathbf{b} = \mathbf{\Phi}_c \mathbf{c}$$

kde $\mathbf{\Phi}_c$ označuje množinu vlastních vektoru. Zřetězené parametry tvaru a textury jsou snadno získatelné díky lineárnímu chování modelu:

(3.27)

$$\mathbf{b} = \begin{pmatrix} \mathbf{W}_s \mathbf{b}_s \\ \mathbf{b}_g \end{pmatrix} = \begin{pmatrix} \mathbf{W}_s \mathbf{\Phi}_s^{\mathrm{T}} (\mathbf{x} - \bar{\mathbf{x}}) \\ \mathbf{\Phi}_g^{\mathrm{T}} (\mathbf{g} - \bar{\mathbf{g}}) \end{pmatrix}$$

Vhodné vyvážení mezi vzdálenostmi pixelů a intenzitami pixelů je uděláno pomocí diagonální matice $\mathbf{W}_s$. Jak získat $\mathbf{W}_s$ je popsáno níže.

Použije se jednoduchá lineární algebra – kompletní instance modelu zahrnující tvar **x** a texturu **g** může být generována za použití parametrů modelu **c**:

(3.28)

$$\mathbf{x} = \bar{\mathbf{x}} + \mathbf{\Phi}_s \mathbf{W}_s^{-1} \mathbf{\Phi}_{c,s}\, \mathbf{c}$$

(3.29)

$$\mathbf{g} = \bar{\mathbf{g}} + \mathbf{\Phi}_g \mathbf{\Phi}_{c,g}\, \mathbf{c}$$

kde

(3.30)

$$\mathbf{\Phi}_c = \begin{pmatrix} \mathbf{\Phi}_{c,s} \\ \mathbf{\Phi}_{c,g} \end{pmatrix}$$

Porovnání pixelové vzdálenosti s intenzitou pixelů

Protože parametry tvaru $\mathbf{b}_s$ jsou v jednotkách pixelové vzdálenosti a parametry textury jsou v jednotkách intenzity pixelů, nebudou si zřejmě odpovídat bez použití vyvážení $\mathbf{W}_s$. Jednoduchá metoda pro odhad $\mathbf{W}_s$ pro získání jednotné váhy je určit poměr r mezi rozsahem všech tvarů a všech textur, které jsou v trénovací množině. Pro připomenutí rozsah parametru b_i se rovná λ_i, pak máme:

(3.31)

$$\mathbf{W}_s = r\mathbf{I} = \begin{bmatrix} r & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & r \end{bmatrix}$$

(3.32)

$$r = \frac{\lambda_g}{\lambda_s} \quad , \quad \lambda_g = \sum \lambda_{g_i} \quad , \quad \lambda_s = \sum \lambda_{s_i}$$

Volba módů variací

Stejně jako v případě PCA pro tvar lze komprimovat kombinovaný model dalším odstraněním nejmenších vlastních módů. To opět zajistí menší citlivost na šum. Pro udržení p procent ze všech kombinovaných variací v trénovací množině by mělo být zvoleno t módů:

(3.33)

$$\sum_{i=1}^{t} \lambda_i \geq \frac{p}{100} \sum_{i=1}^{2n} \lambda_i$$

V této kapitole byl popsán jednotný model vzhledu složený z tvaru a textury. Příklady, které může tento model generovat jsou blízké foto-realistickým obrazům z třídy objektů kterou reprezentuje. Za použití několika málo parametrů může být tento model deformováním tvaru a textury podobný chování, které je pozorováno v trénovací množině.

3.4 Active Appearance Models

Nyní už je vytvořený kombinovaný model vzhledu. Ještě zbývá popsat algoritmus pro vygenerování takové reprezentace modelu, která odpovídá hledanému obrazu obličeje. Tento algoritmus se nazývá AAM hledání. Základem AAM hledání je pojmout hledání jako optimalizační problém, ve kterém rozdíl mezi syntetizovaným objektem vygenerovaným AAM a aktuálním obrazem bude minimalizovaný. Formálně to lze zapsat jako rozdílový vektor $\delta\mathbf{I}$:

(3.34)

$$\delta\mathbf{I} = \mathbf{I}_{image} - \mathbf{I}_{model}$$

Podobnost s obrazem může být zlepšena upravením parametrů modelu a parametrů pozice. Proto následný postup bude založen na normalizovaném texturním vektoru $\delta\mathbf{I}$, který bude označen jako $\delta\mathbf{g}$.

Optimalizace parametrů

Předpokládá se, že prostorový vzor v $\delta\mathbf{g}$ může predikovat upravení modelu a parametrů pozice pro minimalizování $\delta\mathbf{g}$. Nejjednodušší model může být dosažen jako předpoklad lineární závislosti:

(3.35)

$$\delta\mathbf{c} = \mathbf{R}\delta\mathbf{g}$$

Pro určení vhodného **R** z rovnice (3.35) je množina vedených experimentů vložena do multi-variantní lineární regrese.

Každý experiment obsahuje změnu skutečných parametrů známého množství a změřeného rozdílu mezi modelem a části obrazu pod modelem.

Zvažované parametry jsou parametry modelu **c** a parametry pozice **t**. Pro zajištění linearity a rovnováhy kolem nuly je pozice reprezentována jako:

(3.36)

$$t = (s_x, s_y, t_x, t_y)^{\mathrm{T}}$$

kde:

s_x : Kombinace změny velikosti a rotace: $s_x = s\cos\theta - 1$

s_y : Kombinace změny velikosti a rotace: $s_y = s\sin\theta$

t_x : Posunutí ve směru x

t_y : Posunutí ve směru y

V pseudo-kódu může být j-tý experiment vyjádřen jako:

1. Změnit model známým počtem parametrů pozice $\mathbf{t}$ a parametrů modelu $\mathbf{c}$.
2. Aktualizovat současné parametry modelu: $\mathbf{c} = \delta\mathbf{c} + \mathbf{c}_0$.
3. Aktualizovat současnou pozici $\mathbf{t}$, která je dána $\delta\mathbf{t}$ po $\mathbf{t}_0$.
4. Generovat novou instanci modelu generováním nové textury $\mathbf{g}_m$ a nového tvaru $\mathbf{x}$.
5. Získat tvar v obraze $\mathbf{x}_{image}$ zarovnáním tvaru $\mathbf{x}$ na pozici danou $\mathbf{t}$.
6. Sejmout obraz pod $\mathbf{x}_{image}$ a uložit jej do texturního vektoru $\mathbf{g}_{image}$.
7. Normalizovat texturu $\mathbf{g}_{image}$ do $\mathbf{g}_i$.
8. Spočítat normalizovanou rozdílovou texturu: $\delta\mathbf{g} = \mathbf{g}_i - \mathbf{g}_m$.
9. Zapsat $\delta\mathbf{t}$ a $\delta\mathbf{c}$ do experimentálních matic $\mathbf{T}$ a $\mathbf{C}$, každý v j-tém sloupci.
10. Zapsat $\delta\mathbf{g}$ do j-tého sloupce matice $\mathbf{G}$.

Založeno na maticích $\mathbf{T}$, $\mathbf{C}$ a $\mathbf{G}$ a na závislostech:

(3.37)

$$\mathbf{C} = \mathbf{R}_c\mathbf{G} \quad , \quad \mathbf{T} = \mathbf{R}_t\mathbf{G}$$

Multi-variantní regrese pak bude vytvářet závislosti mezi pixelovými rozdíly a změnami parametrů pozice/modelu:

(3.38)

$$\delta\mathbf{c} = \mathbf{R}_c\boldsymbol{\delta}\mathrm{g}$$

(3.39)

$$\boldsymbol{\delta}\mathrm{t} = \mathbf{R}_t\boldsymbol{\delta}\mathrm{g}$$

Základní myšlenkou multi-variantní lineární regrese je předpoklad lineární závislosti mezi dvěma množinami proměnných. V případě AAM je předpokládána lineární závislost mezi naměřenými rozdíly normalizované textury $\delta\mathbf{g}$ a množinou upravených parametrů modelu. Metoda je podrobně rozvedena v [3].

Iterativní optimalizace parametrů

V předchozích kapitolách byl ustanoven pevný základ pro optimalizační proces v AAM. Tato kapitola je zaměřena na detaily použití prediktoru lineární regrese, na němž je založeno AAM hledání.

Princip spočívá v umístění modelu do inicializačního stavu, tedy do referenční pozice a do parametrů modelu $\mathbf{c}_0$ (dáno nějakým dřívějším inicializačním krokem) a získání normalizovaného obrazu pod modelem. Rozdílový vektor $\delta\mathbf{g}$ mezi sejmutým obrazem a instancí modelu může být vypočítán a použit pro predikci množiny parametrů pro vytvoření modelu lépe odpovídajícího obrazu. To je uděláno iterativně, dokud je ve shodě modelu s obrazem pozorováno zlepšení.

Jestliže predikce pro zlepšování shody selže, predikce je buď zesílena, nebo zeslabena po několik dalších iterací a je sledováno, jestli nedojde ke zlepšení.

Procedura zapsaná v pseudo-kódu vypadá následovně:

1. Inicializuj tzv. vektor tlumení $\mathbf{k} = [1.5,0.5,0.25,0.125,0.0625]^{\mathrm{T}}$.
2. Inicializuj model v rámci rozsahu daného predikcí parametrů.
3. Generuj normalizovaný texturní vektor z modelu $\mathbf{g}_m$.
4. Sejmi obraz pod modelem tvaru $\mathbf{x}_{image}$ a ulož jej do $\mathbf{g}_{image}$.
5. Normalizuj $\mathbf{g}_{image}$ do $\mathbf{g}_i$.
6. Vypočítej chybový vektor $\delta\mathbf{g}_0 = \mathbf{g}_i - \mathbf{g}_m$.
7. Vypočítej chybu $E_0 = |\delta\mathbf{g}_0|$.
8. Predikuj změnu v pozici $\delta\mathbf{t} = \mathbf{R}_t\delta\mathbf{g}_0$.
9. Predikuj změnu v parametrech modelu $\delta\mathbf{c} = \mathbf{R}_t\delta\mathbf{g}_0$.
10. Nastav $i = 1$
11. Aktualizuj parametry modelu $\mathbf{c} = \mathbf{c} - k_i\delta\mathbf{c}$.
12. Transformuj tvar pomocí $\delta\mathbf{t}$.
13. Pokračuj kroky 3-7 pro získání nové chyby E_i.
14. Jestliže $E_i > E_0$ nastav $i = i + 1$ a pokračuj krokem 10.
15. Ulož výslednou chybu $E = E_i$.
16. Jestliže nedošlo v poslední iteraci ke zlepšení E, deklaruj konvergenci. Jinak pokračuj krokem 3.

Když algoritmus skončí, tak to neznamená, že byl získán validní model, pouze to indikuje, že díky lineární regresi dochází ke zlepšení.

Hledání by mohlo být značně urychleno hledáním v multi-resolution (pyramidovém) prostoru.

Tato kapitola shrnuje základy AAM. V této a předchozích kapitolách bylo ukázáno, jak odvodit kompaktní statistický model tvaru a textury pomocí PCA. Dále bylo ještě provedeno zkombinování do jednoho sjednoceného modelu a byla popsána efektivní optimalizační metoda založená na multivariantní lineární regresi. Toto vše představuje, co je známo jako Active appearance model.

4 Statistické metody a klasifikace

V této kapitole budou představeny základní statistické metody. Pro hlubší studium je vhodná literatura A. Izenmana [10], kde jsou důkladně probrány všechny nejvýznamnější statistické metody. Kromě metody analýzy hlavních komponent, která je víceméně základní statistickou analýzou pro AAM, budou zmíněny metody vhodné především pro klasifikaci vzorů.

4.1 Analýza hlavních komponent

Z anglického Principal Component Analysis, dále jen jako PCA. PCA slouží pro identifikování vzorů v datech a vyjádření dat cestou zdůraznění podobností a rozdílů. Některé vzory může být v datech o vysoké dimenzi obtížné nalézt, grafická reprezentace není možná, PCA je výkonný nástroj pro analýzu takových dat. Často se také používá ke snižování počtu dimenzí s co nejmenší ztrátou informace. V této kapitole budou nastíněny pouze hlavní charakteristiky metody, pro přesné pochopení je vhodný tutoriál od I. Smitha [2], kde je podrobný postup i s příklady.
Z původních dat je vypočítána kovarianční matice. Z kovarianční matice pak lze vypočítat vlastní vektory a k nim náležící vlastní čísla. Počet vlastní vektorů odpovídá počtu dimenzí původních dat. Vlastní vektory jsou na sebe navzájem kolmé a nesou informaci o vzorech v datech. Vlastní číslo udává velikost vlastního vektoru. Seřadí-li se vlastní vektory podle jejich vlastních čísel, budou komponenty seřazené podle rozptylu. Snížit dimenzi dat lze zachováním pouze těch komponent, jejichž vlastní čísla mohou být pro dané zkoumání zajímavá. Nejčastěji se zachovávají komponenty s největšími vlastními čísly, tedy rozptylem.

4.2 Lineární diskriminační analýza

Nebo také zkráceně LDA. Diskriminace využívá znalosti z trénovací množiny o příslušnosti jednotlivých příkladů k daným třídám. Na základě těchto znalostí konstruuje klasifikátor, který od sebe bude předem definované třídy separovat, tak jak je možné. Samotná klasifikace již neznámé příklady pomocí klasifikátoru predikuje do odpovídajících tříd

Podle [6] je obecně cílem diskriminační analýzy definovat lineární kombinaci z množiny proměnných, která separuje třídy jak je jen možné. Jestliže p jsou proměnné ve vektoru **X**, i-tá diskriminační funkce Z_i je dána:

(4.1)

$$Z_i = a_{i1}X_1 + a_{i2}X_2 + \cdots + a_{ip}X_p$$

Hledání koeficientů a_{ij} je tzv. eigenvalue problémem.

Lineární diskriminační funkcí ve vícerozměrném prostoru jsou vlastní vektory matice $\mathbf{W^{-1}B}$ s odpovídajícími vlastními čísly popisujícími velikost separace, kde **W** a **B** jsou vnitrotřídní a mezitřídní kovarianční matice. Obecný výpočet pro kovarianční matici **B** a **W** dle [5] je následující:

(4.2)

$$\mathbf{B} = \sum_{i=1}^{c} N_i(\mu_i - \mu)(\mu_i - \mu)^T$$

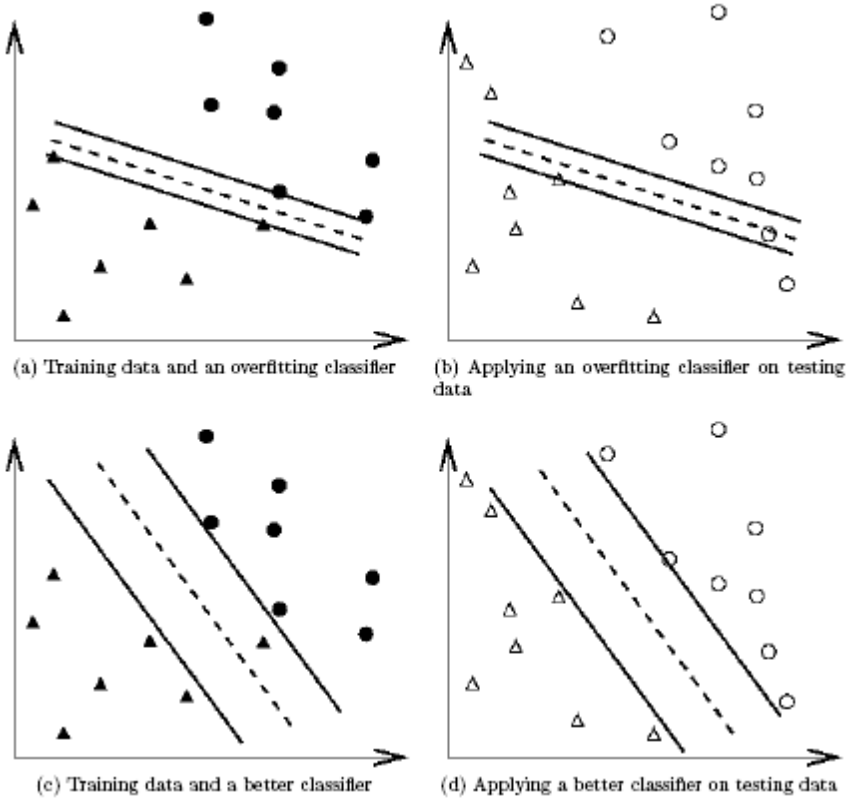
(4.3)

$$\mathbf{W} = \sum_{i=1}^{c} \sum_{x_k \in X_i} (x_k - \mu_i)(x_k - \mu_i)^T$$

kde N_i je počet příkladů v třídě i, c je počet odlišných tříd, μ_i je průměrný vektor příkladů patřících do třídy i, μ je průměrný vektor celé trénovací množiny, X_i reprezentuje příklady patřící do třídy i a x_k je konkrétní příklad.

4.3 SVM

V principu je SVM (support vector machine) binární klasifikátor, který ale lze několika metodami rozšířit na multitřídní klasifikátor. Metody pro multitřídní klasifikaci jsou popsány zde [18]. SVM se natrénuje na trénovacích datech, s označenou příslušností do jednotlivých tříd, tyto data vyhodnotí a určí hranici mezi daty daných tříd. Podrobný princip můžete najít v [17]. Na obr. 4.1 je znázorněn způsob trénování a následné klasifikace pro lineární klasifikátor.



Obr. 4.1: Ukázka použití SVM nástroje. Horní dvojice obrázků ukazuje nevhodnou separaci různých trénovacích dat a následnou klasifikaci. Spodní dvojice obrázků ukazuje již správné separování trénovacích dat a následně i vhodnou klasifikaci (Hsu, Ch., 2010).

5 Přehled existujících metod

Rozpoznávání výrazů tváře je značně rozsáhlá oblast. Zahrnuje lokalizaci obličeje, následnou reprezentaci a klasifikaci získaných příznaků. Existuje mnoho metod pro řešení jednotlivých kroků, ale je nutné podotknout, že žádná z nich není úplně ideální. Rozpoznávání výrazů tváře je dost náročná problematika, především kvůli variabilitě, se kterou se musí potýkat. Problémy způsobuje pořízení obrazu obličeje, úhel snímání, ale i o osvětlení a různé doplňky od pokrývek hlavy přes brýle až třeba po vousy. Každá z metod se vypořádává s těmito problémy po svém, v následujícím textu budou zmíněny nejpoužívanější metody.

5.1 Lokalizace obličeje

Nejčastějšími metodami pro určení polohy obličeje v obraze jsou na pravidlech založené metody, které kódují lidskou znalost typického obličeje. Tyto pravidla zachycují vztahy mezi jednotlivými obličejovými rysy, jako jsou oči, nos, ústa. Patří sem hierarchická metoda shora dolů navržená Yangem a Huangem [13]. Systém obsahuje tři úrovně pravidel. Na nejvyšší úrovni pravidla pouze popisují obecný vzhled obličeje, zatímco na nejnižší úrovni se pravidla zaměřují na detaily obličejových rysů. Hlavním přínosem této metody je hierarchická struktura, díky které dochází ke značné redukci výpočetního času. Nicméně detekční schopnosti nejsou nijak oslnivé.

Opačným přístupem se vyznačuje metoda zdola nahoru, kde se prvně provádí detekce obličejových rysů a až pak se odvozuje přítomnost obličeje. Obličejové rysy jako obočí, oči, nos a ústa jsou extrahovány pomocí hranových detektorů. Z takto extrahovaných rysů je sestaven statistický model pro popis jejich závislostí a ověření přítomnosti obličeje. Jednu takovouto metodu navrhl pro segmentaci obličeje z pozadí pro následnou identifikaci Sirohey [14]. Za pomocí hranové mapy a heuristic jsou odstraněny všechny hrany kromě těch reprezentující konturu obličeje. Tato kontura je pak aproximována elipsou. Další z mnoha takovýchto přístupů je metoda Chetverikova a Lerche [15]. Jejich model obličeje obsahuje 2 černé oblasti reprezentující oči a tři bíle oblasti reprezentující lícní kosti a nos. Model dále obsahuje pruhy pro reprezentování obrysu obličeje, obočí a rtů. Prostorová závislost mezi oblastmi je zakódována pomocí dvou trojúhelníků. Obličej je detekován, pokud jsou nalezeny oblasti v trojúhelníkovém uspořádání s pruhy v jejich okolí. Problémem takto založených metod ale bývá velká citlivost na změny osvětlení, šum apod. Hrany dané stíny totiž mohou být často mnohem výraznější než hrany dané obličejovými rysy.

Pro mnoho aplikací detekce obličeje ale i sledování pohybů rukou se ukázal jako efektivní rys založený na barvě lidské kůže. Ačkoliv různí lidé mají různou barvu kůže, některé studie ukázaly, že většina rozdílů je dána spíše intenzitou než barevnými kanály. Bylo nutné nalézt takový barevný prostor, který co nejlépe pokryje rozložení barev lidské kůže. Barevných modelů je celá řada, nejčastějšími jsou normalizované RGB, HSV a YCrCb. Barevná informace je efektivní nástroj pro detekování obličejových polí a obličejových rysů ale jen v případě když se model barvy kůže může vhodně přizpůsobovat různým světelným podmínkám. Modely barvy kůže nejsou efektivní, když se spektrum zdroje světla výrazně mění. Proto byl McKennem [16] a kolektivem prezentován adaptivní barevný smíšený model pro sledování obličejů za měnících se světelných podmínek. Ovšem barva samotná obvykle není dostačující pro detekování nebo sledování obličejů.

Poslední metoda je od předchozích značně odlišná. Je založena na strojovém učení. Model pro lokalizaci je naučen z trénovací množiny příkladů, které reprezentují variabilitu obličeje. Konkrétně se jedná o detekci objektů pomocí kaskády z jednoduchých příznaků. Jedná se o metodu navrženou

Violou a Jonesem [1], podrobněji popsána v kapitole 2. Metoda se vyznačuje značnou robustností a vysokou rychlostí. Je možné ji natrénovat na lokalizaci téměř libovolných objektů. Metoda tudíž vyžaduje množinu trénovacích obrazů a množinu příznaků.

Pro detailnější představu existujících metod je velmi vhodný podrobný přehled [7].

5.2 Reprezentace obličeje

Používají se metody zaměřené na strukturní vlastnosti obličeje, které jsou invariantní vůči poloze, úhlu pohledu a osvětlení. Tyto metody se označují jako feature-based. Obličej je reprezentován přes popisy rysů a explicitní popis jejich vztahů. Cox [24] pracuje právě s takto založenou detekcí. Rysy jsou v obraze detekovány přes prostorové filtry a odezvy filtrů jsou seskupovány do kategorií buď podle geometrických, nebo jasových vlastností. Vzhledem k definovaným geometrickým vlastnostem obličeje (souměrnost apod.) a pomocí algoritmů feature groupingu je detekována přítomnost obličeje v obraze. Dále se používají různé pravděpodobnostní metody pro vypočítání možnosti, zdali je daný kandidát jistý objekt.

Mezi nejčastěji používané metody patří metody založené na několika standardních vzorech obličeje, které slouží pro popis obličeje jako celku nebo jednotlivých obličejových rysů. Jedná se o tzv. template-based metody. Pro detekci jsou počítány korelace mezi vstupním obrazem a uloženými vzory. Jedná se o jeden z nejúspěšnějších přístupů pro objektovou representaci, je založen na analýze hlavních komponent (PCA) jako například eigenface přístup. Jeho slabší stránkou je citlivost na perspektivní deformace a změny osvětlení. PCA aproximuje pouze textury, geometrickou informaci nepočítá. Další přístup založený na PCA je AAM (Active Appearance Model), viz kapitola3. Tento přístup rozšiřuje eigenface přístup zahrnutím geometrické informace. Výhoda tohoto přístupu je zdatnost modelace téměř každého obličeje, gest apod. Obecně eigenface přístupy kódují informaci založenou na pixelech.

Na rozdíl od template-based metody, kde vzory jsou předdefinovány expertem, u metody appearance-based vzory vznikají učením z příkladů v obrazech. Obecně tyto metody spoléhají na techniky statistické analýzy a strojového učení pro hledání charakteristik obrazů s obličeji a bez obličejů. Naučené charakteristiky jsou ve tvaru distribučních modelů nebo diskriminačních funkcí, které jsou následně použity pro detekci obličeje. Mezitím je uskutečněna redukce kvůli výpočetní efektivitě a detekční účinnosti.

Kombinací přístupů feature-based a template-based je dán tzv. bunch graf, který navrhnul Wiskott [25]. Metoda je založena na diskrétní vlnkové transformaci. Síť Gáborových vlnek je aplikována na množinu ručně vybraných bodů, tak, že každý bod je representován množinou odezev filtru. Během fáze trénování jsou body počítány na množině trénovacích obrazů, ale vždy pro stejné množiny význačných bodů. Po fázi trénování existuje celá množina odezev filtru pro každý význačný bod. Množina odezev filtru dohromady s relativní pozicí bodu je definována jako bunch graf. Za účelem representace nového obrazu bunch graf prvně automaticky hledá význačné body za pomocí sady uložených bodů a to pro každý bod.

Poslední metodou je tzv. histogram-based. Objekty jsou popsány a charakterizovány vektory lokálně naměřených vlastností, jako je barva, derivace, z těch se následně sestavují histogramy. Vícerozměrné histogramy jsou použity pro aproximování funkce hustoty rozložení pravděpodobnosti pro lokální a globální příznaky v obrazu. Histogram nepoužívá geometrickou informaci. Nadto ještě segmentace popředí/pozadí není v této representaci možná a přístup pak není dostatečně robustní vůči různorodosti pozadí, když není segmentace řádně provedena.

5.3 Klasifikace výrazů

Support vector machine (SVM) je efektivní metoda, pro rozpoznávání vzorů. Hledá takovou hyper-rovinu, která separuje co nejvíce bodů stejné třídy, zatímco maximalizuje vzdálenost jednotlivých tříd od hyper-roviny [17]. Ovšem SVM je binární klasifikátor a pro klasifikaci výrazů obličeje je zcela jistě potřeba více tříd. Existují dvě strategie rozšíření na více tříd. První strategie tzv. jeden proti všem zahrnuje natrénování n SVM, kde každé SVM separuje danou třídu ze zbytku trénovací množiny. Druhá strategie je založena na dvojici klasifikátorů, zahrnující trénování různých SVM pro separování každé dvojice tříd. Srovnání multi-třídních metod je v [18].

Skryté Markovovy modely (Hidden Markov Models - HMM) [19] jsou rozšířením teorie Markovových řetězců, kde daný výstup je pravděpodobnostní funkce stavu. Identifikace je dosažena výběrem takového HMM, které dosahuje nejvyšší pravděpodobnosti. Rychlá výpočetní konvergence dělá HMM prakticky použitelné pro zpracování v reálném čase.

Dalším klasifikačním nástrojem jsou neuronové sítě s vícevrstvým perceptronem (Multilayer Perceptron - MLP) [20]. Hledá se přijatelné lokální minimum ve váhovém prostoru pro dosažení minimální chyby. Váhy jsou upravovány pomocí algoritmu zpětného šíření, který je vlastní trénovací procedurou. Během trénování MLP sestaví separační hyperplochy ve vstupním prostoru. Po natrénování může MLP použít získané schopnosti pro rozpoznávání výrazů. MLP je efektivní nástroj pro rozpoznávání výrazů, ale kvůli vykonávání PCA analýzy je dosti pomalý.

Pro zvýšení rychlosti rozpoznávání lze použít metodu založenou na bodové korelaci v kombinaci s AAM modelem [21]. Výsledkem AAM jsou body popisující významné geometrické rysy obličeje. Spočítají se Euklidovské vzdálenosti mezi jednotlivými významnými body a na základě závislostí mezi jednotlivými vzdálenostmi lze určit výraz obličeje. Rychlost metody je vykoupena přesností a proto se hodí spíše do mobilních platforem, například robotů.

Další metody vycházející z AAM jsou zmíněny v přehledu [22].

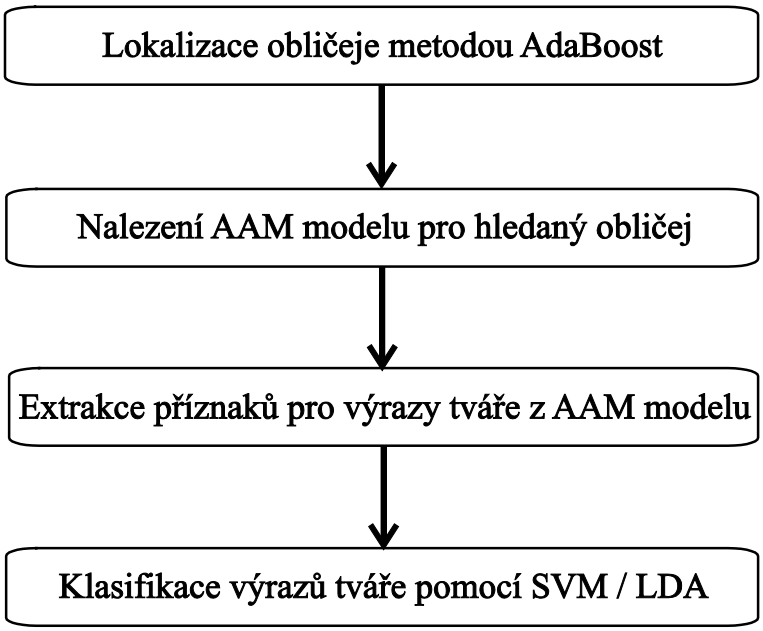
6 Návrh rozpoznávače výrazu tváře

Po rozsáhlejší teorii přichází na řadu vlastní návrh. Budou zužitkovány probrané postupy, důležitá bude především část zabývající se tvorbou AAM modelu (kapitola 3). Budou následovat základní myšlenky, jak by mohl kompletní rozpoznávač výrazů tváře pracovat.

Návrh robustního rozpoznávače je značně složitá úloha. Musí se zohlednit řada problémů spojených především s variabilitou lidského obličeje. Už jen při lokalizaci obličeje nastává řada problémů kvůli úhlu pořízení obličeje, světelným podmínkám apod.

Podobně jako ve většině systémů pro rozpoznávání obličejových charakteristik i ve mnou navrženém systému je rozpoznávač tvořen třemi základními bloky. První blok představuje nalezení oblasti obrazu, která zahrnuje hledaný obličej. Obdélníkový region s obličejem je podroben dalšímu zpracování, jedná se již o nalezení příznaků obličeje odpovídající jednotlivým výrazům. Tyto příznaky jsou následně klasifikovány do tříd obličejových výrazů.

Základní myšlenku rozpoznávače již máme, nyní můžeme jednotlivé bloky nahradit konkrétními algoritmy. Lokalizace obličeje je řešena metodou AdaBoost kaskády s jednoduchými příznaky. Lokalizovaný obličej je reprezentován pomocí AAM modelu, tedy nalezením odpovídající instance AAM modelu. Z modelu jsou vybrány vhodné parametry, které slouží jako příznaky pro jednotlivé výrazy tváře. Vyextrahované příznaky se vhodně natrénovaným klasifikátorem klasifikují do známých tříd. Pro lepší pochopení celé struktury je na obr. 6.2 znázorněno schéma celého systému. Nyní bude následovat konkrétní popis zmíněných prvků rozpoznávače.



Obr. 6.1: Finální struktura rozpoznávače výrazu tváře.

6.1 Lokalizace obličeje

Cílem je nalézt alespoň přibližně oblast zájmu, ve které se vyskytuje obličej. Metoda AAM, o které bude řeč v další kapitolce, sice rovněž umí určit polohu obličeje v obraze, ale to slouží pouze pro upřesnění pozice. Inicializační oblast musí alespoň z části pokrývat hledaný obličej, aby byl nalezen

odpovídající model. Pro lokalizaci obličeje je použita velmi rychlá a robustní metoda AdaBoost kaskády s jednoduchými příznaky (viz kapitola 2). Realizace této metody je součástí dostupné AAM knihovny.

Vstupem je libovolný, ať už šedotónový či barevný obrázek s jediným obličejem. Jednoduchou úpravou se mi podařilo lokalizaci rozšířit na libovolný počet obličejů v obraze. Pro každý obličej je nalezen čtvercový region a ten se stává samostatným vstupem pro další zpracování. Příklad úspěšné lokalizace je zobrazen na obr. 6.2.

Obr. 6.2: Lokalizace obličeje metodou AdaBoost.

6.2 Nalezení AAM modelu

Lokalizační metodou AdaBoost byly nalezeny oblasti obrazu, které mohou obsahovat obličej. Ovšem takováto oblast neobsahuje žádné další informace o obličeji. Pro nalezení informací popisující hledaný obličej a pro zpřesnění polohy obličeje se použije metoda AAM (viz kapitola 3). Pro mé účely jsem použil opět knihovnu AAM. Jen pro připomenutí, AAM může modelovat jakékoliv objekty, na které se natrénuje. AAM byl natrénován na obrazech obličejů z přímého pohledu. 58 význačných bodů, které určují geometrii obličeje, byly zadány manuálně.

Po natrénování modelu se provede tzv. optimalizace modelu, tedy nalezení odpovídající instance modelu pro hledaný obličej. K tomu se nabízejí dvě metody, obě součástí knihovny. Jedná se o metodu fixního jakobiánu (Fixed Jacobian - FJ) a inverzní kompoziční metodu (Invesre Compositional - IC) [26]. Princip FJ je již zmíněn v kapitole 3, pouze zdůrazním, že metoda FJ je sice o něco pomalejší, ale výsledná optimalizace je úspěšnější (viz obr. 6.3).

Obr. 6.3: Optimalizace metodou FJ (vlevo) a metodou IC (vpravo).

Ačkoliv se zdá, že takto zvolené body dostatečně přesně popisují tvar jednotlivých prvků obličeje, pro informaci o výrazu obličeje už tyto tvarové body tak vhodné nejsou. Pro definování výrazu člověka je důležitý především tvar úst, zda-li jsou otevřená, zda-li jsou vidět zuby, poloha, tvar obočí a tvar očí. Ze zvolených význačných bodů vůbec nelze rozpoznat, zda-li jsou ústa otevřena. Další nedostatečnou informací je poloha obočí, lze sice určit vzdálenost od očí, ale chybí poloha vůči temeni hlavy. Naproti tomu některé body jsou pro vyjádření výrazu zbytečné. Není potřeba tak přesně kopírovat dolní čelist a tvar nosu. Tímto se dostávám k novému popisu tvaru, který obsahuje 71 význačných bodů, lze vidět na obr. 6.4. Díky popisu celé hlavy lze dosáhnout lepší detekce a navíc lze pracovat i s informacemi z oblasti čela, jako jsou například vrásky, které mohou být rovněž užitečné pro rozpoznávání výrazu tváře.

Výsledek optimalizace pro nově zvolené body má ovšem v některých případech problém se správným nalezením polohy a tvaru úst. Menší nepřesnost je ostatně vidět i na obr. 6.4, kde především koutky úst nejsou přesně nalezeny. I když je AAM model natrénovaný pro různé varianty úsměvů obličeje, dochází k této drobné odchylce poměrně často. Jenom na základě tvaru, tedy bez texturní informace, by mohlo být obtížné takovýto tvar úst klasifikovat jako úsměv. Řešením by mohlo být opětovné snížení počtu význačných bodů kolem úst. Se zvyšujícím se počtem bodů totiž narůstá variabilita modelu, tudíž je obtížnější nalézt odpovídající interpretaci. Úplným vyloučením

vnitřních bodů úst, které modelují otevřená ústa, sice klesne úspěšnost detekce výrazu podle tvaru, ale tvarovou informaci by šlo zkombinovat s texturou v oblasti úst, to ale bude podrobněji zmíněno až v následující kapitolce.

Obr. 6.4: Nalezení AAM modelu s upraveným tvarem.

Doposud byly řešeny pouze vstupní data pro nalezení modelu. Nyní k samotnému procesu optimalizace modelu pro hledaný obličej. Tady nastával problém s některými typy obličejů, které nebyly dostatečně zastoupeny v trénovacích datech, model začal v iterativním procesu optimalizace přetékat přes obličej. Kvůli tomu byla provedena menší úprava původního procesu. V prvním kroku iterace je vytvořena obdélníková obálka pro vygenerovaný model obličeje. Nalezený model obličeje je pak v každém dalším kroku iterace kontrolován, zda-li nepřesahuje přes obálku o více, než je zvolený koeficient.

Kvalita nalezeného modelu je samozřejmě ovlivněna počtem iterací procesu optimalizace. Jako vhodné se ukázalo zvolit 30 iterací. Jedná se kompromis mezi kvalitou a výpočetním časem. Při větším počtu už jsou rozdíly v kvalitě výsledků minimální.

Existuje řada možností na vylepšení detekční schopnosti modelu. Jedním může být zkombinovat proces nalezení modelu s lokalizační metodou. Jak již bylo zmíněno, kromě lokalizace celého obličeje lze metodu AdaBoost použít i pro lokalizaci očí popřípadě úst. Tyto další regiony by sloužily pro nastavení inicializační polohy bodů úst a očí.

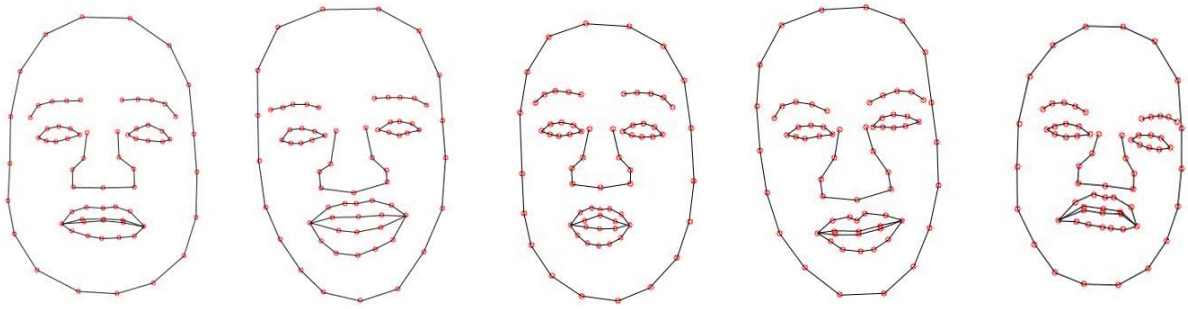
Výsledkem procesu optimalizace je vygenerovaný model odpovídající hledanému obličeji. Model je popsán parametry. V případě FJ metody se jedná o kombinované parametry, které současně generují tvar i texturu modelu. V případě IC metody je model generován pouze parametry tvaru, textura se mění až podle změněného tvaru.

6.3 Extrakce příznaků

Nalezená reprezentace AAM modelu nese komplexní informace o hledaném obličeji. Z modelu lze vyčíst přesný tvar obličeje, ale i texturní vlastnosti obličeje. Záleží jen na nás, jak a které informace z modelu použijeme jako příznaky pro klasifikaci výrazu tváře.

Geometrie obličeje

Není problém z AAM modelu vyextrahovat body popisující přesný tvar obličeje. Jsou dva způsoby, buď z parametrů modelu vygenerovat body tvaru odpovídající význačným bodům, nebo jako příznak použít přímo parametry modelu IC metody. Ovšem jak je vidět na obr. 6.5, geometrie nemusí být dostatečná pro rozpoznání jednotlivých výrazů. Geometrie pro některé typy výrazů se liší jen minimálně. To ostatně dokazují i výsledky v kapitole 8.3. Jako příznak pro rozpoznávání výrazu lze použít pouze geometrii očí, úst a obočí. Tvar hlavy ani nosu se významně nemění se změnou výrazu obličeje.

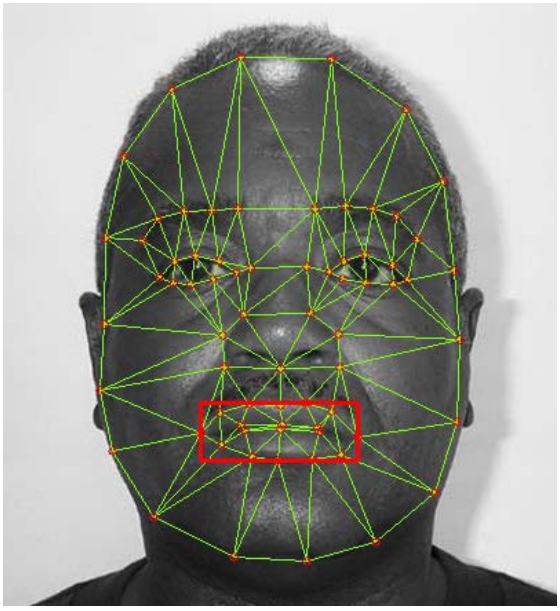


Obr. 6.5: Geometrie obličeje pro neutrální, šťastný, překvapený, mrzutý a smutný výraz.

Geometrie a textura obličeje

Jak bylo zmíněno výše, čistě geometrická informace nemusí být dostatečná pro popis jednotlivých výrazů. Jako vhodné se jeví zkombinovat geometrické vlastnosti s texturními. Nejjednodušším způsobem je přímé použití parametrů modelu FJ metody. Tyto parametry v sobě kódují jak geometrické tak texturní informace. Takto zvolené příznaky již umožňují lepší klasifikaci, ale pořád je problém z takto zvolených příznaků rozlišit mezi neutrálním a smutným výrazem. Důvodem je, že několik málo parametrů kóduje tvar a texturu celého obličeje.

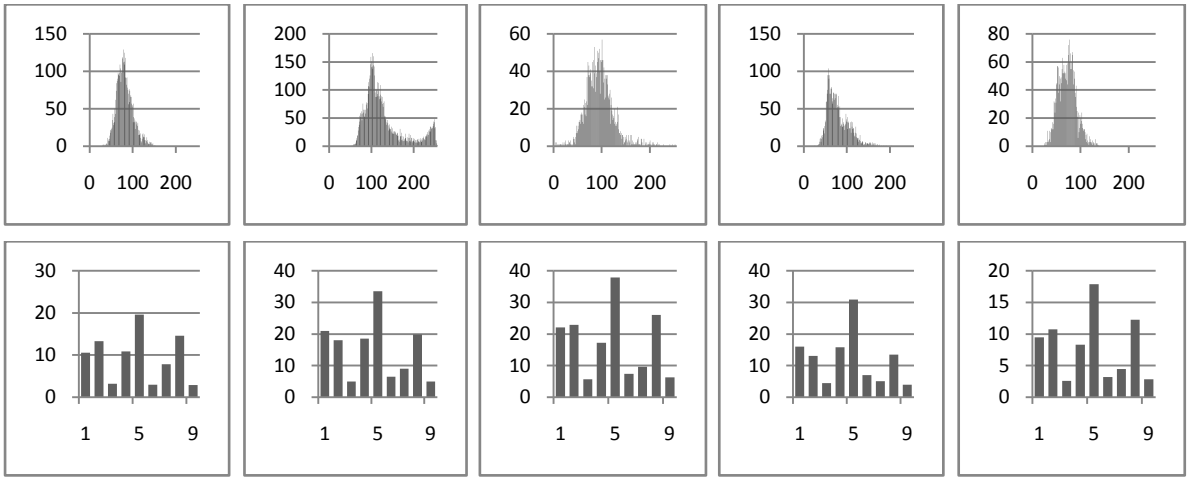
Možným vylepšením stávajícího přístupu extrakce příznaků je geometrickou informaci zkombinovat s vlastním texturním příznakem. Jako geometrická informace by se daly použít body tvaru zmíněné výše. A z míst v obličeji, kde geometrická informace není dostatečná pro rozpoznání výrazu vzít zvlášť texturní informaci. Nabízí se především oblast úst. Není problém tuto oblast přesně definovat na základě bodů popisující ústa. Výsledkem by mohl být obdélníkový region opisující tvar úst (viz obr. 6.6). Nabízí se otázka, jaké příznaky z této texturní oblasti extrahovat. Chceme, aby příznak rozlišoval zavřená ústa, otevřená ústa a zuby.



Obr. 6.6: AAM model s detekcí oblasti úst.

Nejjednodušším řešením se zdá být pracovat s jasovou informací. Tedy sestavit histogram intenzit pro sejmutou oblast úst. Na obr. 6.7 jsou typické tvary úst pro jednotlivé výrazy tváře. Na obr. 6.8 jsou zobrazeny histogramy intenzit úst pro odpovídající výrazy. Jelikož histogram pracuje s jasovou informací, nemůže pokrýt rozdíly, které jsou způsobené tvarem úst. Jak je vidět na obrázcích, histogram od sebe rozliší pouze otevřená, zavřená ústa a viditelné zuby. Ale zkombinováním s tvarovou informaci, by přeci jen mohlo dojít ke zlepšení klasifikace. Ale použít celý histogram jako příznakový vektor není příliš vhodné, obsahoval by 256 hodnot, což by pro následnou klasifikaci mohlo být výpočetně náročné ale i méně přesné. Vhodné by bylo histogram jistým způsobem aproximovat, pojmout základní tvar histogramu, například adaptivním průměrováním a nalezením lokálních maxim a minim. Ještě před tím by se měl histogram normalizovat.

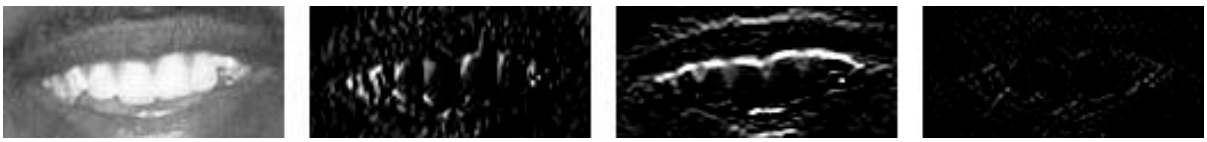
Obr. 6.7: Ústa odpovídající neutrálnímu, šťastnému, překvapenému, mrzutému a smutnému výrazu.



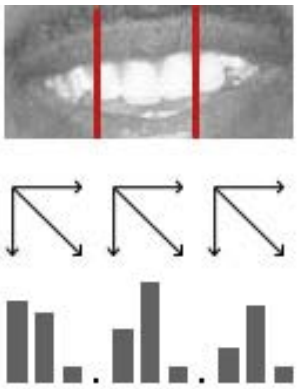
Obr. 6.8: Histogram intenzit (nahoře) a histogram orientovaných gradientů (dole) pro jednotlivé výrazy úst.

Způsobem pro extrakci příznaků z textury úst, který reprezentuje i tvar úst je histogram orientovaných gradientů [28]. Jedná se o jednoduchou metodu, která se používá například pro detekci lidských postav, pro rozpoznávání obličejů a nebo jen pro detekci úsměvu. Na vstupní obraz se aplikuje detektor hran pomocí Sobelova operátoru. Je sestaven histogram, který má tolik binů (košů), kolik směrů v obraze se bude určovat. Hodnota daného binu představuje přehled o zastoupení gradientů v daném směru.

Konkrétně byl na vstupní obraz aplikován detektor hran se sobelovým operátorem $3x3$ postupně ve směru osy x, osy y a ve směru xy (viz obr. 6.9). Výsledné obrazy zvýraznily hrany v těchto 3 základních směrech. Hodnota vzniklá sumou hodnot každého obrazu a následným normalizováním udává příslušnost k danému směru. Vzniklý histogram bude mít právě 3 hodnoty pro 3 pozorované směry. Takovýto příznak asi nebude dostatečně přesně charakterizovat celá ústa. Proto se obraz úst rozdělí na tři stejně velké buňky a každá buňka se bude analyzovat samostatně. Histogramy z každé buňky se skonkatenují a výsledný příznak bude mít již 9 hodnot (obr. 6.10). Histogramy pro typické výrazu úst a srovnání s původními histogramy intenzit je na obr. 6.8.



Obr. 6.9: Příklad určení hran v obraze. Vlevo původní obraz, detekce hran ve směru x, detekce hran ve směru y, detekce hran ve směru xy.



Obr. 6.10: Příklad určení orientace gradientů v obraze. Nahoře obraz rozdělený na buňky, určení směru gradientu v každé buňce, reprezentace obrazu pomocí konkatenovaných histogramů jednotlivých buněk.

6.4 Klasifikace výrazů tváře

Před samotnou klasifikací se musí klasifikátor natrénovat. Trénovací sadou bude množinu 40-ti obrazů pro každou třídu výrazů. Nalezne se model pro každý obraz a následné extrahování příznaků podle postupu výše. Příznak bude tvořen vektorem parametrů modelu a vektorem z histogramu orientovaných gradientů. S příznakem se bude zacházet jako s jedním vektorem. Výsledkem je pro každou známou třídu výrazů množina 40-ti vektorů. Podle známých příznaků klasifikátor nalezne separaci prostoru, podle které pak bude provádět klasifikaci. Ve své práci jsem navrhnul jednoduchý klasifikátor založený na LDA a pro porovnání úspěšnosti klasifikace jsem využil nástroje SVM.

LDA

V části trénování se vytvoří průměrné vektory příznaků z jednotlivých tříd výrazů. Následně se sestaví kovarianční matice reprezentující chování uvnitř jednotlivých tříd výrazů:

(6.1)

$$\mathbf{W} = \sum_{i=1}^{c} \sum_{x_k \in X_i} (x_k - \mu_i)(x_k - \mu_i)^T$$

kde c je počet tříd výrazů, μ_i je průměrný vektor příznaků patřící do třídy i, X_i reprezentuje vektory příznaků patřící do třídy i a x_k je vektor příznaků konkrétního příkladu.

Samotná klasifikace neznámého vektoru x se provádí jako Mahalanobisova vzdálenost mezi dvěma vektory:

(6.2)

$$D_i = (x - \mu_i)\mathbf{W^{-1}}(x - \mu_i)$$

Určuje vzdálenost mezi i-tou třídou a hledaným vektorem x. Pomocí inverzní vnitrotřídní matice $\mathbf{W^{-1}}$ dochází k potlačení variací uvnitř jednotlivých tříd výrazů.

SVM

Trénování klasifikátoru ovlivňuje celá řada parametrů, bude nás zajímat především typ SVM a typ jádra. Pro námi zvolené příznaky se jako vhodné jeví použít typ jádra pro lineární separaci a typ SVM pro multi-třídní klasifikaci z ohraničených problémů.

Výsledný natrénovaný model se pak použije v predikci pro klasifikaci nového vektoru.

7 Implementace rozpoznávače výrazu

V této kapitole budou představeny použité nástroje a ukázána strukturu celého rozpoznávače výrazů lidské tváře. Jelikož jádro rozpoznávače tvoří metoda AAM převzatá z GNU knihovny Active Appearance Models Library (AAMLibrary), nemá význam procházet implementaci této metody, akorát zmíním některé možné úpravy. Dále se zaměřím na to, jak jsou jednotlivé části systému propojeny.

7.1 Použité nástroje

Rozpoznávač výrazů tváře byl implementován v programovacím jazyce C++ ve vývojovém prostředí Visual Studio 2008 na operačním systému MS Windows 7. Pro prácí s obrazem je využíváno open source knihovny OpenCV. Pro metodu AAM byla využita knihovna AAMLibrary, která obsahuje řadu metod pro kompletní návrh AAM. Klasifikace byla provedena pomocí SVM nástroje, konkrétně se jednalo o knihovnu BSVM.

OpenCV

OpenCV je volně šiřitelná multiplatformní knihovna pro zpracování obrazu a videa. I když nedávno byla vydána verze 2.1, musel jsem zůstat u verze 1.0. Knihovna AAMLibrary, rovněž využívá funkcí OpenCV, ale byla vyvíjena pro verzi 1.0. S verzí 2.0 je knihovna AAMLibrary nekompatibilní. Pro OpenCV verze 1.1 se sice knihovnu AAMLibrary podařilo zkompilovat, ale docházelo k nestandardnímu chování výsledného modelu. Tyto problémy se mi bohužel nepodařilo odhalit, tudíž jsem nakonec zvolil OpenCV 1.0, pod kterým bylo AAMLibrary vyvíjeno.

Všechny funkce OpenCV jsou srozumitelně popsány v literatuře Learning OpenCV [9].

AAMLibrary

Jedná se GNU knihovnu pro kompletní realizaci AAM metody. Zahrnuje trénování a optimalizaci modelu. Obsahuje dvě metody pro optimalizaci modelu, metodu FJ (fixed jacobian) a metodu IC (inverse compositional). Součástí implementace je i lokalizace obličeje. Knihovna je navržena jak pro prácí se statickým obrazem, tak i s videem. Pro mé účely jsem knihovnu omezil pouze pro statický snímek.

K lokalizaci obličeje AAMLibrary využívá funkcí knihovny OpenCV. Součástí OpenCV knihovny jsou natrénované kaskády příznaků pro jednotlivé typy objektů. V přístupu navrženém v AAMLibrary je pro lokalizaci použita kaskáda pro obličeje z předního pohledu. Kromě kaskád pro obličeje jsou součástí OpenCV i kaskády pro lokalizaci očí. Pro snížení falešné detekce lze přidat k lokalizaci obličeje právě lokalizaci očí. Za region obsahující obličej by se považovala pouze ta oblast, kde by oba lokalizátory byly úspěšné. Stačilo by kontrolovat, zda-li regiony očí náleží do regionu celého obličeje.

BSVM

Existuje celá řada knihoven pro realizaci SVM. Zvolil jsem knihovnu BSVM, která vychází z LibSVM. Jedná se o volně šiřitelnou knihovnu. Je možné ji použít v binární podobě nebo jak v mém případě volat její funkce přímo z programu. Tato knihovna podporuje celou řadu nastavení.

7.2 Struktura programu

Program Build

Jedná se o první část rozpoznávače. Jak vyplývá z názvu, program sestaví z trénovací množiny obrazů se zadanou geometrií AAM model pro další zpracování. Výstupem je natrénovaný AAM model.

Program Lda

Ve druhé části rozpoznávače se načte AAM model a natrénovaná kaskáda pro lokalizaci obličeje. Postupně pro každý obraz z trénovací sady se provede lokalizace obličeje. Pro lokalizovaný obličej se nalezne AAM model. Vyextrahované příznaky modelu se použijí jako vstup pro natrénování klasifikátoru, kterým je buď zvolen LDA nebo SVM. Výstupem je natrénovaný model pro zvolený klasifikátor.

Program Fit

V tomto programu se již provádí samotná klasifikace. Načte se AAM model a natrénovaná kaskáda pro lokalizaci obličeje. Postupně pro každý obraz z trénovací sady se opět provede lokalizace obličeje. Pro lokalizovaný obličej se nalezne AAM model. Vyextrahované příznaky jsou metodou klasifikace zvolenou v předchozím kroku klasifikovány do pěti tříd odpovídající jednotlivým výrazům tváře. Výstupem je třída, do které byl hledaný obličej klasifikován.

8 Experimentální výsledky

Na nejdůležitějších komponentách detektoru obličejových rysů, jako je AAM trénování, AAM hledání a klasifikace výrazů tváře byla provedena série jednoduchých testů. V této kapitole se pokusím na základě testů vystihnout chování a případné problémy jednotlivých částí. Testování bude zaměřeno především na robustnost přesnost a okrajově i na rychlost jednotlivých komponent.

Ovšem testy časové náročnosti nelze považovat za příliš objektivní, protože rychlost zpracování je závislá zejména na použitém hardwaru ale i operačním systému. Pro testování byla použita PC sestava s procesorem Pentium Dual-Core E5200 o pracovní frekvenci 2,50 GHz s operačním systémem MS Windows 7.

8.1 Trénování modelu

Metoda pro sestavení modelu, tedy AAM podle Stegmanna [3] patří mezi časově nejnáročnější úkony celého detekčního systému. Proto se v tomto případě zaměřím na testy časové náročnosti. Pro dosažení dostatečně přesné reprezentace AAM modelem je důležité jaká se zvolí trénovací množina. Jako nejvhodnější se jeví trénovací sada o 32 obrázcích. Testování bylo provedeno na obličejích s vyznačeným tvarem pomocí 58 a 71 bodů. Pro tuto testovací sadu budou sestaveny dva modely, modely založené na metodě FJ a metodě IC. Jak je vidět v tab. 7.1, časová náročnost metody FJ je podstatně vyšší, ale to není důvod, proč tuto metodu opomenout, jak bude ukázáno v dalším testování, časová náročnost je vykoupena lepší optimalizací modelu.

Metoda	Počet bodů modelu	Čas sestavení modelu [s]
FJ	58	285,3
FJ	71	389,2
IC	58	4,6
IC	71	4,5

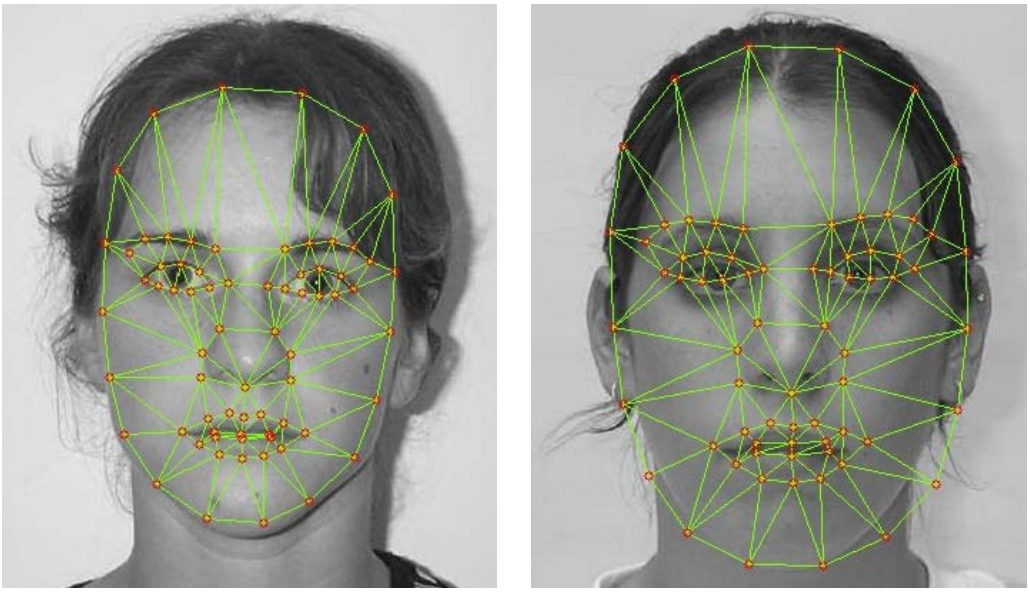
Tab. 8.1: Čas sestavení AAM modelu.

8.2 Optimalizace parametrů

Předpoklad úspěšné klasifikace výrazů je dostatečně přesná reprezentace obličeje modelem. Budeme testovat dvě varianty AAM optimalizace a to FJ a IC pro dva typy trénovacích obrazů. Jako testovací sada se použije kompletní databáze A lifespan database of adult facial stimuli [27], která obsahuje 1020 obrázků. Kromě času nalezení modelu nás bude zajímat především přesnost výsledného modelu. Ke zjištění chyby optimalizace se využije rozdílu mezi hledaným obličejem a syntetizovaným obličejem. Z tab. 7.2 lze vyčíst, že přesnější reprezentace dosahuje FJ metoda. Nejlepších výsledků bylo dosaženo pro model tvořený 71 body, který pokrývá tvar celé hlavy. Příklady optimalizace modelem jsou na obr. 8.1.

Metoda	Počet bodů modelu	Čas nalezení modelu [ms]	Chyba optimalizace
FJ	58	304,6	1,895
FJ	71	279,3	1,497
IC	58	95,3	1,862
IC	71	101,3	2,141

Tab. 8.2: Čas a chyba optimalizace AAM modelu.



Obr. 8.1: Příklad úspěšné (vlevo) a neúspěšné optimalizace pro 71 bodů.

8.3 Klasifikace výrazu tváře

Klasifikace výrazů tváře byla provedena na stejné obličejové databázi jako testování AAM optimalizace [27]. Část databáze byla použita pro natrénování klasifikátoru, pro každou třídu výrazů byla použita sada 40-ti obrazů. Jako testovací sada pro klasifikátor byla použita zbývající část databáze (viz tab. 8.3).

Šťastný	Neutrální	Smutný	Překvapený	Celkem
218	540	24	38	820

Tab. 8.3: Počet testovacích obrázků pro jednotlivé třídy výrazů.

Nalezený AAM model obsahuje celistvé informace o hledaném obličeji, jednotlivé informace použijeme jako příznak pro klasifikaci výrazů a budeme ověřovat úspěšnost. Kromě různých příznaků budeme klasifikaci provádět na základě různých modelů. Bude se jednat o modely založené na metodách FJ a IC a to s 58 a se 71 body popisující tvar. Samotná klasifikace je založena na metodách LDA a SVM.

První testování bylo založené na metodě optimalizace FJ s původním počtem bodů tvaru. Jako příznak byly použity parametry modelu. Tyto parametry generují jak tvar, tak i texturu. Jak je vidět z tab. 8.4, výsledky nejsou nijak oslnivé, především pak pro neutrální třídu výrazů. Důvodem je nejspíše to, že 58 zvolených bodů nedokáže dostatečně přesně reprezentovat ústa, zda-li jsou otevřená, zavřená a podobně.

Klasifikátor	Šťastný	Neutrální	Smutný	překvapený	Celkem
LDA	59%	24%	50%	40%	35%
SVM	50%	16%	58%	40%	27%

Tab. 8.4: Úspěšnost klasifikace pro metodu FJ s 58 body, příznakem jsou parametry modelu.

Podobně jako v prvním případě, opět se bude jednat o model založený na metodě FJ s 58 body. Tentokrát se jako příznak použije vygenerovaný výsledný tvar (tab. 8.5). Ani geometrická informace se neukázala jako dostatečný příznak pro klasifikaci výrazu. Takto zadané body tvaru prostě nejsou schopny rozlišit jednotlivé výrazy.

Klasifikátor	Šťastný	Neutrální	Smutný	překvapený	Celkem
LDA	59%	24%	50%	40%	35%
SVM	16%	82%	75%	0%	60%

Tab. 8.5: : Úspěšnost klasifikace pro metodu FJ s 58 body, příznakem jsou body tvaru.

I když metoda IC, co se týká do přesnosti, se neukázala jako příliš vhodná, provedeme pouze jednu sérii testů, jen pro srovnání s metodou FJ. Jako příznakový vektor byly použity parametry modelu, ale na rozdíl od FJ metody, parametry metody IC generují jen tvar. Z výsledků v tab. 8.6 je úspěšnost dle očekávání nižší než u metody FJ, což je hlavně způsobenou horší reprezentací obličeje modelem. Pro další testování se proto zaměřím pouze na metodu FJ.

Klasifikátor	Šťastný	Neutrální	Smutný	překvapený	Celkem
LDA	26%	34%	33%	37%	32%
SVM	41%	2%	0%	0%	12%

Tab. 8.6: : Úspěšnost klasifikace pro metodu IC s 58 body, příznakem jsou parametry modelu.

Následně se zaměřím na sledování výsledků z modelu, který je popsán 71 body. Jak je vidět v tab. 8.2, měl by tento model dosahovat lepší reprezentace a tudíž i následná klasifikace by měla být úspěšnější. Zase vybereme jako příznak postupně parametry modelu, tedy texturu s tvarem, poté tvar samotný (viz tab. 8.7 a tab. 8.8). Podle tab. 8.7 došlo k již značnému zvýšení úspěšnosti, výjimkou je třída výrazů „neutrální“, která je chybně klasifikována převážně do třídy „smutný“. To je pravděpodobně způsobeno nepřesným nalezením tvaru v oblasti úst (viz obr. 8.1 vpravo). Klasifikace pouze na základě tvaru (viz tab. 8.8) dosahuje pro LDA stejných výsledků jako v předchozím případě, ale podstatně horších výsledků v případě SVM. SVM pravděpodobně nedokáže tyto příznaky dobře separovat. Problémem může být pro SVM nedostatečná velikost trénovací sady.

Klasifikátor	Šťastný	Neutrální	Smutný	překvapený	Celkem
LDA	52%	35%	46%	50%	41%
SVM	46%	27%	58%	34%	33%

Tab. 8.7: Úspěšnost klasifikace pro metodu FJ se 71 body, příznakem jsou parametry modelu.

Klasifikátor	Šťastný	Neutrální	Smutný	překvapený	Celkem
LDA	52%	35%	46%	50%	41%
SVM	23%	6%	17%	5%	11%

Tab. 8.8: : Úspěšnost klasifikace pro metodu FJ se 71 body, příznakem jsou body tvaru.

Výsledky z předchozích dvou tabulek nicméně můžeme považovat za dobrý základ pro další úpravy. Zatím jsme pracovali pouze s informace z AAM modelu, možné zlepšení by přineslo zkombinování s dalším příznakem. Jelikož hlavní problém je v rozpoznání tvaru úst, přidá se příznak z textury v oblasti úst. Bude se jednat o již zmíněný histogram směrových gradientů. Klasifikace podle příznaku tvořeného body tvaru a histogramem je v tab. 8.9. Jedná se o základní variantu histogramu s třemi buňkami. V případě LDA došlo opět ke zlepšení, úspěšnost již dosahuje 50%, to je způsobeno dostatečně kvalitním příznakem reprezentující ústa. V případě SVM přetrvává problém s nesprávnou separací příznaků.

Klasifikátor	Šťastný	Neutrální	Smutný	překvapený	Celkem
LDA	49%	56%	42%	55%	54%
SVM	26%	21%	58%	5%	23%

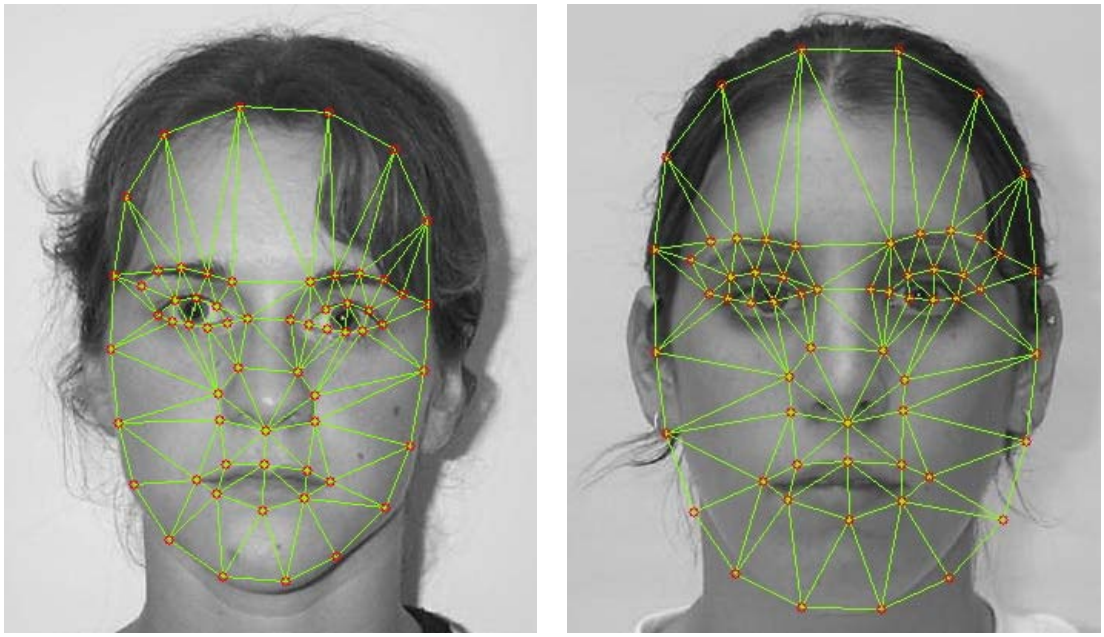
Tab. 8.9: : Úspěšnost klasifikace pro metodu FJ se 71 body, příznakem jsou body tvaru a histogram orientovaných gradientů.

Zkusme histogram orientovaných gradientů zpřesnit na 6 buněk pro lepší reprezentaci úst a jako další příznak použít parametry modelu. V tab. 8.10 můžeme pozorovat, mírné zlepšení pro LDA a konečně i výrazné zlepšení pro SVM.

Klasifikátor	Šťastný	Neutrální	Smutný	překvapený	Celkem
LDA	50%	57%	46%	55%	55%
SVM	55%	52%	54%	29%	52%

Tab. 8.10: Úspěšnost klasifikace pro metodu FJ se 71 body, příznakem jsou parametry modelu a histogram orientovaných gradientů.

Protože nalezení bodů tvaru kolem úst v případě modelu se 71 body nemusí dostatečně přesné, nabízí se poslední úprava. Jednoduše odstranit některé body úst, zachovat pouze body pro obrys úst. Již není potřeba reprezentovat otevřená ústa, jelikož pro ústa je použit stávající příznak z histogramu orientovaných gradientů. Výsledkem je model popsaný 61 body (viz obr. 8.2). Z tab. 8.11 pozorujeme další zlepšení, které je způsobeno pouze tím, že reprezentace modelem se 61 body je přesnější. Tuto variantu můžeme považovat za nejúspěšnější, úspěšnost klasifikace se blíží šedesáti procentům.



Obr. 8.2: Příklad úspěšné (vlevo) a neúspěšné optimalizace pro 61 bodů.

Klasifikátor	Šťastný	Neutrální	Smutný	překvapený	Celkem
LDA	57%	59%	42%	66%	58%
SVM	56%	57%	50%	34%	56%

Tab. 8.11: : Úspěšnost klasifikace pro metodu FJ se 61 body, příznakem jsou parametry modelu a histogram orientovaných gradientů.

V tab. 8.12 a v tab. 8.13 je ukázáno chování klasifikátoru LDA a SVM pro všechny třídy výrazů. Z dat lze vyčíst, na kterých třídách klasifikace nejvíce selhává.

	Mrzutý	Šťastný	Neutrální	Smutný	Překvapený
Šťastný	2%	57%	32%	0%	9%
Neutrální	14%	2%	59%	15%	10%
Smutný	54%	0%	0%	42%	4%
překvapený	18%	0%	3%	13%	66%

Tab. 8.12: LDA klasifikace do jednotlivých tříd výrazů.

	Mrzutý	Šťastný	Neutrální	Smutný	Překvapený
Šťastný	0%	56%	21%	0%	22%
Neutrální	10%	9%	57%	8%	16%
Smutný	38%	4%	4%	50%	4%
překvapený	26%	3%	0%	37%	34%

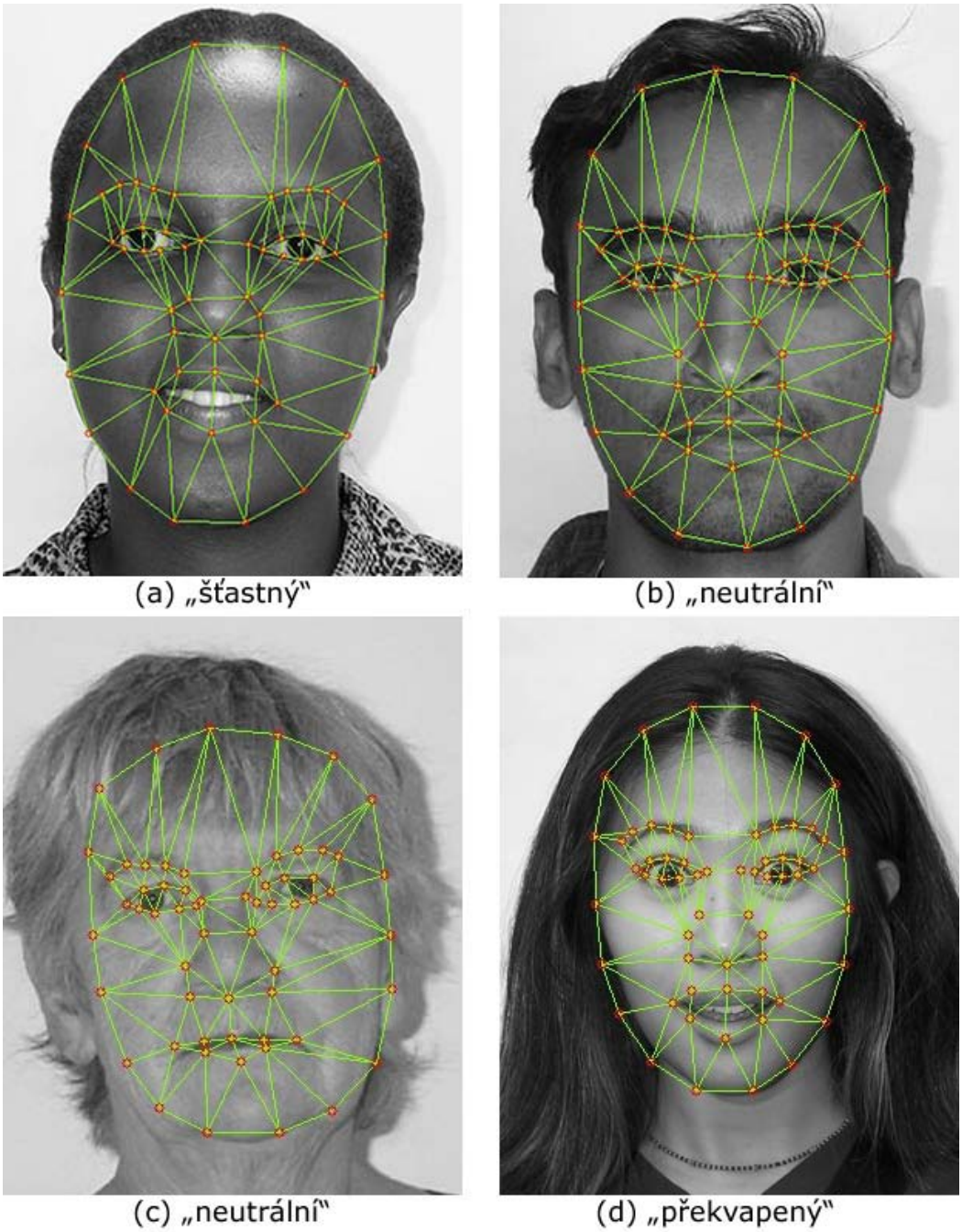
Tab. 8.13: SVM klasifikace do jednotlivých tříd výrazů.

Na závěr už jen ověříme kvalitu natrénování klasifikátoru na trénovacích datech (tab. 8.14), tedy 40 obrázků pro každou třídu výrazů. Výsledky jsou přijatelné, ke zlepšení především u SVM by byla potřeba větší trénovací sada.

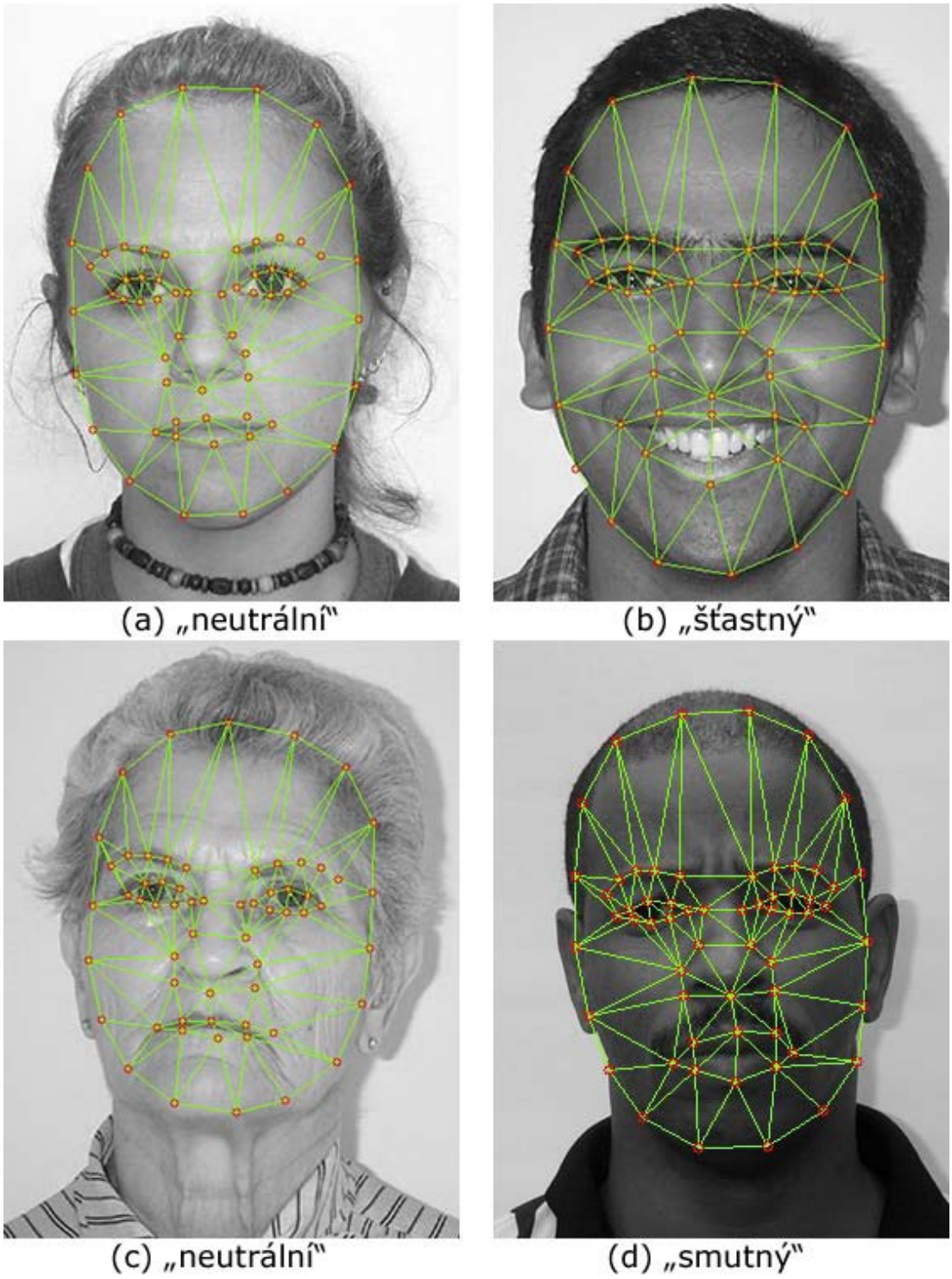
Klasifikátor	Mrzutý	Šťastný	Neutrální	Smutný	překvapený	Celkem
LDA	85%	97%	98%	80%	87%	89%
SVM	75%	100%	95%	77%	84%	86%

Tab. 8.14: Ověření natrénovaných klasifikátorů.

Pro poslední, nejúspěšnější verzi klasifikátoru uvedu reálné výsledky. Na obr. 8.3 jsou zobrazeny osoby se správně rozpoznanými výrazy. Na obr. 8.4 jsou osoby, u kterých rozpoznání výrazu nebylo úspěšné. U obličejů (a), (b) a (d) je špatná klasifikace pochopitelná. „Neutrální“ výraz může být podobný výrazu „smutný“, výraz „smutný“ může být podobný výrazu „mrzutý“ apod. V případě obličeje (c), je chybná klasifikace dosti neočekávaná, výraz „překvapený“ je spíše zaměnitelný s výrazem „šťastný“. Chyba může být způsobena chybně vypočteným příznakem z textury v oblasti úst. Oblast pro získání textury je pravděpodobně příliš malá.



Obr. 8.3: Příklad úspěšné klasifikace výrazů tváře.



Obr. 8.4: Příklad neúspěšné klasifikace výrazů tváře. Obličej (a) klasifikovaný jako „smutný“, (b) jako „překvapený“, (c) jako „překvapený“, (d) jako „mrzutý“.

9 Závěr

Cílem mé práce bylo navrhnout komplexní rozpoznávač výrazů tváře. Systém, který bude obličej lokalizovat, následně vhodně reprezentovat a klasifikovat do tříd známých výrazů tváře. Hlavní požadavek byl kladen především na robustnost vůči variabilitě výrazu různých osob. Jak jistě čtenář tuší, nejedná se o jednoduchý úkol a v současné době žádná s používaných metod není zdaleka stoprocentní. Přesto se pokusím těmto požadavkům alespoň přiblížit.

Pro detekci obličeje jsem zvolil možnost reprezentace obličeje modelem. Jako vhodná se jevila metoda AAM, která dovede velmi přesně modelovat téměř libovolný obličej. Jedná se o velmi robustní metodu použitelnou pro téměř libovolné objekty, což závisí na zvolené trénovací množině. Jedná se o jádro celého systému a proto je ji věnována celá teorie v kapitole 3. Pro správnou funkčnost AAM bylo třeba využít některou z metod lokalizace obličeje, která definuje oblast zájmu pro následnou reprezentaci modelem. Jako lokalizační metodu jsem zvolil takovou metodu, která je schopna ve velmi krátkém čase nalézt obličejové oblasti nezávisle na natočení obličeje a světelných podmínkách. Jedná se o metody strojového učení založené na AdaBoost. AAM model nese komplexní informace o hledaném obličeji. Je potřeba z modelu vybrat takové příznaky, které budou dostatečně přesně reprezentovat daný výraz hledaného obličeje. Příznaky se dále klasifikují metodou LDA i metodou SVM.

Ve své práci jsem AAM metodu převzal z otevřené knihovny AAMLibrary, provedl jsem pouze několik modifikací. Především jsem se zaměřil na klasifikaci výrazů tváře založené na AAM modelu. Velká část práce byla věnována experimentování s různými variantami modelu a s výběrem vhodných příznaků pro klasifikaci.

Předpoklad pro úspěšnou klasifikaci je dostatečně přesná reprezentace AAM modelem pro hledaný obličej. Největší přesnosti reprezentace bylo dosaženo s upraveným modelem, který byl definován pomocí 71 bodů tvaru. Jako příznak pro klasifikaci výrazu tváře byly zvoleny parametry modelu, tedy parametry kódující jak texturu, tak tvar. Celková úspěšnost klasifikace dosáhla 55%. Přidáním texturního příznaku pro oblast úst a zrušením některých bodů tvaru kolem úst se celková úspěšnost klasifikace zvýšila až na 58%. Těchto hodnot bylo dosáhnuto s klasifikací pomocí LDA. V případě SVM byla úspěšnost řádově v procentech nižší. To je ale pravděpodobně způsobeno malou trénovací sadou.

Lepších výsledků by se dalo dosáhnout použitím jiné metody optimalizace modelu, robustnější metody, která by byla schopná realizovat větší variabilnost obličejů. Kromě úst hlavně oči a obočí charakterizují výraz lidské tváře, bylo by tedy vhodné přidat texturní příznaky pro oblasti kolem očí. Ke zlepšení by pravděpodobně vedlo i upravení použitého texturního příznaku pro ústa tak, aby byl invariantní vůči natočení obličeje.

Rozpoznávání výrazu tváře je zatím realizováno pouze pro statický snímek. Nicméně rozšíření na video-sekvence by vyžadovalo realtime zpracování. K potřebnému zrychlení může pomoct použití pyramidového prostoru, tedy víceúrovňová architektura s různými rozlišeními obrazu. Dalším opatřením pro zvýšení výkonnosti by mohlo být snížení přesnosti modelu, tedy počet parametrů, kterým je model obličeje reprezentován. Je nutné stanovit hranici, kde dochází k přínosnému potlačení šumu a kde už dochází ke ztrátě důležitých rysů.

Literatura

[1] Viola, P., Jones, M.: Rapid Object Detection using a Boosted Cascade of Simple. Cambridge, 2001.

[2] Smith, L.: A tutorial on Principal Components Analysis. 2002.

[3] Stegmann, M.: Active Appearance Models. Lyngby, Technical University of Denmark, 2000.

[4] Edwards, G., Taylor, C., Cootes, T.: Learning to Identify and Track Faces in Image Sequences. Manchaster, University of Manchaster, 2003.

[5] Xiaoguang, L.: Image Analysis for Face Recognition. East Lansing, Michigan State University.

[6] Edwards, G., Lanitis, A., Cootes, T.: Statistical Models of Face Images - Improving Specificity. Manchaster, University of Manchaster.

[7] Yang, M., Kriegman, D., Ahuja, N.: Detecting Faces in Images: A Survey. IEEE Transaction on Pattern Aanalysis and Machine Intelligence, 2002.

[8] Cootes, T., Taylor, C.: Statistical Models of Appearance for Computer Vision. Manchaster, University of Manchaster, 2000.

[9] Bradski, G., Kaebler, A.: Learning OpenCV. Computer Vision with the OpenCV Library. 2008.

[10] Izenman, A., J.: Modern Multivariate Statistical Techniques. Regression, Classification and Manifold Learning. Department of Statistics, Temple University, Speakman Hall, Philadelphia. 2008.

[11] Lanitis, A., Taylor, C., Cootes, F.: Automatic Interpretation and Coding of Face Images Using Flexible Models. IEEE Transaction on Pattern Aanalysis and Machine Intelligence, 1997.

[12] Rueckert, D., Thomaz, C.: Estimation of Within-Class Matrix in Image Classification. London, 2008.

[13] Yang G., Huang, T.: Human Face Detection in a Complex Background. Pattern Recognition, Volume 27, 1994.

[14] Sirohey, S.: Human Face Segmentation and Indentification. Computer Vision Laboratory, Center of Automotion Research, University of Maryland, 1993.

[15] Chetverikov, D., Lerch, A.: Multiresolution Face Detection.Theoretical Foundations of Computer Vision, Volume 69, 1993.

[16] McKenna, S., Raja, Y., Gong, S.: Tracking Colour Objects Using Adaptive Mixture Models. Image and Vision Computing, Volume 17, 1998.

[17] Bousquet, O., Boucheron, S., Lugosi, G.: Introduction to Statistical Learning Theory.

[18] Hsu, Ch., Lin, Ch.: A Comparison of Methods for Multi-Class Support Vector Machines.Department of Computer Science and Information Engineering, National Taiwan University.

[19] Lien, J.: Automatic Recognition of Facial Expression Using Hidden Markov Models and Estimation of Expression Intensity. The Robotics Institue, Carnegie Mellon University, Pittsburgh, 1998.

[20] Padgett, C., Cottrell, G., Adolphs, R.: Categorical perception in Facial Emotion Classification. Proceedings of the 18th Annual Cognitive Science Conference, San Diego, 1996.

[21] Duc, T., Huu, T., Tan, L.: Facial Expression Recognition Using AAM Algorithm. Division of Automatic Control, Ho Chi Minh University of Technology, Vietnam.

[22] Abboud, B., Davoine, F., Dang, M.: Facial Expression Recognition and Synthesis Based on an Appearance Model. Heudiasyc Laboratory CNRS, University of Technology of Compiegne, France, 2004.

[23] Hsu, Ch., Chang, Ch., Lin, Ch.: Practical Guide to Support Vector Classfification. Department of Computer Science, National Taiwan University, Taiwan, 2010.

[24] Cox, I., Ghosn, J., Yianilos, P.: Feature-Based Face Recognition Using Mixture-Distance. Department of Computer Science, University of Montreal, 1995.

[25] L. Wiskott: Face recognition by elastic bunch graph matching. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1997.

[26] Asthana, A., Saragih, J, Wagner, M., Goecke R.: Evaluating AAM Fitting Methods for Facial Expression Recognition. 2009.

[27] Minear, M., Park, D.: A lifespan database of adult facial stimuli. Behavior Research Methods, Instruments, & Computers. 36, 630-633, 2004.

[28] Bai, Y., Guo, L., Jin, L., Huang, Q.: A Novel Feature Extraction Method Using Pyramid Histogram of Orientation Gradients for Smile Recognition. School of Electronic and Information, South China University of Technology, China.

A Příloha

A.1 Popis programu Build

První část rozpoznávače, která sestaví z trénovací množiny obrazů se zadanou geometrií AAM model pro další zpracování.

Spuštění programu

Program se neinstaluje, je spustitelný přímo z DVD. Spouští se s následujícími parametry:

Build Type Level TrainSet DetectFace AAMModel

- Type – metoda AAM optimalizace, 0 – FJ, 1 - IC
- Level - počet úrovní pyramidového prostoru.
- TrainSet – adresář s trénovacími daty, která jsou ve formátu JPG pro texturu a DAT pro tvar.
- DetectFace - soubor pro detekci obličeje z předního pohledu.
- AAMModel – výstupní AAM model.

Příklad spuštění programu:

Build -t 0 -p 1 ../TrainImage jpg dat haarcascade_frontalface_alt.xml model.amf

Ovšem není nutné modely sestavovat, hotové modely jsou již uloženy na DVD.

A.2 Popis programu Lda

Druhá část rozpoznávače, která vytvoří z trénovací sady klasifikátor výrazů tváře.

Spuštění programu

Program se neinstaluje, je spustitelný přímo z DVD. Spouští se s následujícími parametry:

Lda TrainImage jpg DetectFace AAMModel ClassModel Train TypeClass

- TrainImage – adresáře s trénovacími obrazy pro jednotlivé výrazy, jsou ve formátu JPG.
- DetectFace - soubor pro detekci obličeje z předního pohledu.
- AAMModel – vstupní AAMmodel pro následnou optimalizaci.
- ClassModel – výstupní natrénovaný model klasifikátoru.
- Train – výstupní soubor s extrahovanými příznaky pro trénovací data.
- TypeClass – typ zvolené klasifikace, LDA nebo SVM.

Příklad spuštění programu:

Lda ../ Annoyed ../ /Happy .. /Neutral .. /Sad ../Surprise jpg haarcascade_frontalface_alt.xml model.amf svm.amf train.dat svm

Modely klasifikátorů jsou již také uloženy na DVD.

A.3 Popis programu Fit

Poslední část provádí samotnou klasifikaci výrazů z testovací sady.

Spuštění programu

Program se neinstaluje, je spustitelný přímo z DVD. Spouští se s následujícími parametry:

Fit Image DetectFace AAMModel ClassModel Test Class TypeClass

- Image – adresář s testovacími obrazy, která jsou ve formátu JPG.
- DetectFace - soubor pro detekci obličeje z předního pohledu.
- AAMModel – vstupní AAMmodel pro následnou optimalizaci.
- ClassModel – vstupní natrénovaný model klasifikátoru.
- Test – výstupní soubor s extrahovanými příznaky pro testovací data.
- Class – výstupní soubor s třídami výrazů tváře pro jednotlivé testovací obrazy.
- TypeClass – typ zvolené klasifikace, LDA nebo SVM.

Příklad spuštění programu:

Fit ../TestImage jpg haarcascade_frontalface_alt.xml model.amf svm.amf test.dat class svm